Character Language Models for Chinese Word Segmentation and Named Entity Recognition

Bob Carpenter
Alias-i, Inc.
carp@alias-i.com

Abstract

We describe the application of the LingPipe toolkit to Chinese word segmentation and named entity recognition for the 3rd SIGHAN bakeoff.

1 Word Segmentation

Chinese is written without spaces between words. For the word segmentation task, four training corpora were provided with one sentence per line and a single space character between words. Test data consisted of Chinese text, one sentence per line, without spaces between words. The task is to insert single space characters between the words. For this task and named entity recognition, we used the UTF8-encoded Unicode versions of the corpora converted from their native formats by the bakeoff organizers.

2 Named Entity Recognition

Named entities consist of proper noun mentions of persons (`PER`), locations (`LOC`), and organizations (`ORG`). Two training corpora were provided. Each line consists of a single character, a single space character, and then a tag. The tags were in the standard BIO (begin/in/out) encoding. `B-PER` tags the first character in a person entity, `I-PER` tags subsequent characters in a person, and `0` characters not part of entities. We segmented the data into sentences by taking Unicode character `0x3002`, which is rendered as a baseline-aligned small circle, as marking end of sentence (`EOS`). As judged by our own sentence numbers (see Figures 1 and 2), this missed around 20% of the sentence boundaries in the City U NE corpus and 5% of the boundaries in the Microsoft NE corpus. Test data is in the same format as the word segmentation task.

3 LingPipe

LingPipe is a Java-based natural language processing toolkit distributed with source code by Alias-i (2006). For this bakeoff, we used two LingPipe packages, `com.aliasi.spell` for Chinese word segmentation and `com.aliasi.chunk` for named-entity extraction. Both of these depend on the character language modeling package `com.aliasi.lm`, and the chunker also depends on the hidden Markov model package `com.alias.hmm`. The experiments reported in this paper were carried out in May 2006 using (a prerelease version of) LingPipe 2.3.0.

3.1 LingPipe's Character Language Models

LingPipe provides n-gram based character language models with a generalized form of Witten-Bell smoothing, which performed better than other approaches to smoothing in extensive English trials (Carpenter 2005). Language models provide a probability distribution $P(\sigma)$ defined for strings $\sigma \in \Sigma^*$ over a fixed alphabet of characters Σ. We begin with Markovian language models normalized as random processes. This means the sum of the probabilities for strings of a fixed length is 1.0.

The chain rule factors $P(\sigma c) = P(\sigma) \cdot P(c|\sigma)$ for a character c and string σ. The n-gram Markovian assumption restricts the context to the previous $n-1$ characters, taking $P(c_n|\sigma c_1 \cdots c_{n-1}) = P(c_n|c_1 \cdots c_{n-1})$.

The maximum likelihood estimator for n-grams is $\hat{P}_{\text{ML}}(c|\sigma) = \text{count}(\sigma c)/\text{extCount}(\sigma)$, where $\text{count}(\sigma)$ is the number of times the sequence σ was observed in the training data and $\text{extCount}(\sigma)$

Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, pages 169–172,
Sydney, July 2006.

is the number of single-character extensions of σ observed: $\text{extCount}(\sigma) = \sum_c \text{count}(\sigma c)$.

Witten-Bell smoothing uses linear interpolation to form a mixture model of all orders of maximum likelihood estimates down to the uniform estimate $P_U(c) = 1/|\Sigma|$. The interpolation ratio $\lambda(d\sigma)$ ranges between 0 and 1 depending on the context:

$$\begin{aligned}\hat{P}(c|d\sigma) &= \lambda(d\sigma)P_{\text{ML}}(c|d\sigma) \\ &+ (1-\lambda(d\sigma))\hat{P}(c|\sigma) \\ \hat{P}(c) &= \lambda()P_{\text{ML}}(c) \\ &+ (1-\lambda())(1/|\Sigma|)\end{aligned}$$

Generalized Witten-Bell smoothing defines the interpolation ratio with a hyperparameter θ:

$$\lambda(\sigma) = \frac{\text{extCount}(\sigma)}{\text{extCount}(\sigma) + \theta \cdot \text{numExts}(\sigma)}$$

We take $\text{numExts}(\sigma) = |\{c|\text{count}(\sigma c) > 0\}|$ to be the number of different symbols observed following σ in the training data. The original Witten-Bell estimator set the hyperparameter $\theta = 1$. LingPipe's default sets θ equal to the n-gram order.

3.2 Noisy Channel Spelling Correction

LingPipe performs spelling correction with a noisy-channel model. A noisy-channel model consists of a source model $P_s(\mu)$ defining the probability of message μ, coupled with a channel model $P_c(\sigma|\mu)$ defining the likelihood of a signal σ given a message μ. In LingPipe, the source model P_s is a character language model. The channel model P_c is a (probabilistically normalized) weighted edit distance (with transposition). LingPipe's decoder finds the most likely message μ to have produced a signal σ: $\text{argmax}_\mu P(\mu|\sigma) = \text{argmax}_\mu P(\mu) \cdot P(\sigma|\mu)$.

For spelling correction, the channel $P_c(\sigma|\mu)$ is a model of what is likely to be typed given an intended message. Uniform models work fairly well and ones tuned to brainos and typos work even better. The source model is typically estimated from a corpus of ordinary text.

For Chinese word segmentation, the source model is trained over the corpus with spaces inserted. The noisy channel deterministically eliminates spaces so that $P_c(\sigma|\mu) = 1.0$ if σ is identical to μ with all of the spaces removed, and 0.0 otherwise. This channel is easily implemented as a weighted edit distance where deletion of a single space is 100% likely (log probability edit "cost" is zero) and matching a character is 100% likely, with any other operation being 0% likely (infinite cost). This makes any segmentation equally likely according to the channel model, reducing decoding to finding the highest likelihood hypothesis consisting of the test string with spaces inserted. This approach reduces to the cross-entropy/compression-based approach of (Teahan et al. 2000). Experiments showed that skewing these space-insertion/matching probabilities reduces decoding accuracy.

3.3 LingPipe's Named Entity Recognition

LingPipe 2.1 introduced a hidden Markov model interface with several decoders: first-best (Viterbi), n-best (Viterbi forward, A* backward with exact Viterbi estimates), and confidence-based (forward-backward).

LingPipe 2.2 introduced a chunking implementation that codes a chunking problem as an HMM tagging problem using a refinement of the standard BIO coding. The refinement both introduces context and greatly simplifies confidence estimation over the approach using standard BIO coding in (Culotta and McCallum 2004). The tags are `B-`*T* for the first character in a multi-character entity of type *T*, `M-`*T* for a middle character in a multi-character entity, `E-`*T* for the end character in a multi-character entity, and `W-`*T* for a single character entity. The out tags are similarly contextualized, with additional information on the start/end tags to model their context. Specifically, the tags used are `B-O-`*T* for a character not in an entity following an entity of type T, `I-O` for any middle character not in an entity, and `E-O-`*T* for a character not in an entity but preceding a character in an entity of type *T*, and finally, `W-O-`*T* for a character that is a single character between two entities, the following entity being of type *T*. Finally, the first tag is conditioned on the begin-of-sentence tag (*BOS*) and after the last tag, the end-of-sentence tag (*EOS*) is generated. Thus the probabilities normalize to model string/tag joint probabilities.

In the HMM implementation considered here, transitions between states (tags) in the HMM are modeled by a maximum likelihood estimate over the training data. Tag emissions are generated by bounded character language models. Rather than the process estimate $P(X)$, we use $P(X\#|\#)$, where $\#$ is a distinguished boundary character

Corpus	*Encod*	*Sents*	*Chars*	*Uniq*	*Words*	*Uniq*	*Test S*	*Test Ch*	*Unseen*
City U HK	HKSCS (trad)	57K	4.3M	5113	1.6M	76K	7.5K	364K	0.046%
Microsoft	gb18030 (simp)	46K	3.4M	4768	1.3M	63K	4.4K	173K	0.046%
Ac Sinica	Big5 (trad)	709K	13.2M	6123	5.5M	146K	11.0K	146K	0.560%
Penn/Colo	CP936 (simp)	19K	1.3M	4294	0.5M	37K	5.1K	256K	0.160%

Figure 1: Word Segmentation Corpora

Corpus	*Sents*	*Chars*	*Uniq*	*LOC*	*PER*	*ORG*	*Test S*	*Test Ch*	*Unseen*
City U HK	48K	2.7M	5113	48.2K	36.4K	27.8K	7.5K	364K	0.046%
Microsoft	44K	2.2M	4791	36.9K	17.6K	20.6K	4.4K	173K	0.046%

Figure 2: Named Entity Recognition Corpora

not in the training or test character sets. We also train with boundaries. For Chinese at the character level, this bounding is irrelevant as all tokens are length 1, so probabilities are already normalized and there is no contextual position to take account of within a token. In the more usual word-tokenized case, it normalizes probabilities over all strings and accounts for the special status of prefixes and suffixes (e.g. capitalization, inflection).

Consider the chunking consisting of the string *John J. Smith lives in Seattle.* with *John J. Smith* a person mention and *Seattle* a location mention. In the coded HMM model, the joint estimate is:

$\hat{P}_{\text{ML}}(\texttt{B-PER}|\texttt{BOS}) \cdot \hat{P}_{\texttt{B-PER}}(John\#|\#)$
$\cdot \hat{P}_{\text{ML}}(\texttt{I-PER}|\texttt{B-PER}) \cdot \hat{P}_{\texttt{I-PER}}(J\#|\#)$
$\cdot \hat{P}_{\text{ML}}(\texttt{I-PER}|\texttt{I-PER}) \cdot \hat{P}_{\texttt{I-PER}}(.\#|\#)$
$\cdot \hat{P}_{\text{ML}}(\texttt{E-PER}|\texttt{I-PER}) \cdot \hat{P}_{\texttt{E-PER}}(Smith\#|\#)$
$\cdot \hat{P}_{\text{ML}}(\texttt{B-O-PER}|\texttt{E-PER}) \cdot \hat{P}_{\texttt{B-O-PER}}(lives\#|\#)$
$\cdot \hat{P}_{\text{ML}}(\texttt{E-O-LOC}|\texttt{B-O-PER}) \cdot \hat{P}_{\texttt{E-O-LOC}}(in\#|\#)$
$\cdot \hat{P}_{\text{ML}}(\texttt{W-LOC}|\texttt{E-O-LOC}) \cdot \hat{P}_{\texttt{W-LOC}}(Seattle\#|\#)$
$\cdot \hat{P}_{\text{ML}}(\texttt{W-O-EOS}|\texttt{W-LOC}) \cdot \hat{P}_{\texttt{W-O-EOS}}(.\#|\#)$
$\cdot \hat{P}_{\text{ML}}(\texttt{EOS}|\texttt{W-O-EOS})$

LingPipe 2.3 introduced an n-best chunking implementation that adapts an underlying n-best chunker via rescoring. In rescoring, each of these outputs is scored on its own and the new best output is returned. The rescoring model is a longer-distance generative model that produces alternating out/entity tags for all characters. The joint probability of the specified chunking is:

$\hat{P}_{\texttt{OUT}}(c_{\texttt{PER}}|c_{\texttt{BOS}})$
$\cdot \hat{P}_{\texttt{PER}}(John\ J.\ Smith c_{\texttt{OUT}}|c_{\texttt{OUT}})$
$\cdot \hat{P}_{\texttt{OUT}}(\ lives\ in\ c_{\texttt{LOC}}|c_{\texttt{PER}})$
$\cdot \hat{P}_{\texttt{LOC}}(Seattle c_{\texttt{OUT}}|c_{\texttt{OUT}})$
$\cdot \hat{P}_{\texttt{OUT}}(.c_{\texttt{EOS}}|c_{\texttt{LOC}})$

where each estimator is a character language model, and where the c_T are distinct characters not in the training/test sets that encode begin-of-sentence (BOS), end-of-sentence (EOS), and type (e.g. PER, LOC, ORG). In words, we generate an alternating sequence of OUT and type estimates, starting and ending with an OUT estimate. We begin by conditioning on the begin-of-sentence tag. Because the first character is in an entity, we do not generate any text, but rather generate a character indicating that we are done generating the OUT characters and ready to switch to generating person characters. We then generate the phrase *John J. Smith* in the person model; note that type estimates always begin and end with the $c_{\texttt{OUT}}$ character, essentially making them bounded models. After generating the name and the character to end the entity, we revert to generating more out characters, starting from a person and ending with a location. Note that we are generating the phrase *lives in* including the preceding and following space. All such spaces are generated in the OUT models for English; there are no spaces in the Chinese input. Next, we generate the location phrase the same way as the person phrase. Next, we generate the final period in the OUT model and then the end-of-sentence symbol. Note that the OUT category's language model shoulders the brunt of the burden of estimating contextual effects. It conditions on the preceding type, so that the likelihood of *lives in* is conditioned on following a person entity. Furthermore, the choice to begin an entity of type location is based on the fact that it follows *lives in*. This includes begin-of-sentence and end-of-sentence effects, so the model is sensitive to initial capitalization in the out model as a distribution of character sequences likely to follow BOS. Similarly, the

Corpus	R	P	F_1	*Best* F_1	*OOV*	R_{OOV}
City Uni Hong Kong	.966	.957	.961	.972	4.0%	.555
Microsoft Research	.959	.955	.957	.963	3.4%	.494
Academia Sinica	.951	.935	.943	.958	4.2%	.389
U Penn and U Colorado	.919	.895	.907	.933	8.8%	.459

Figure 3: Word Segmentation Results (Closed Category)

Corpus	R	P	F_1	*Best* F_1	P_{LOC}	R_{LOC}	P_{PER}	R_{PER}	P_{ORG}	R_{ORG}
City Uni HK	.8417	.8690	.8551	.8903	.8961	.8762	.8749	.8943	.6997	.8176
MS Research	.8097	.8188	.8142	.8651	.8351	.8716	.7968	.8438	.7739	.6899

Figure 4: Named Entity Recognition Results (Closed Category)

end-of-sentence is conditioned on the preceding text, in this case a single period. The resulting model defines a (properly normalized) joint probability distribution over chunkings.

4 Held-out Parameter Tuning

We ran preliminary tests on MUC 6 English and City University of Hong Kong data for Chinese and found baseline performance around 72% and rescored performance around 82%. The underlying model was designed to have good recall in generating hypotheses. Over 99% of the MUC test sentences had their correct analysis in a 1024-best list generated by the underlying model. Nevertheless, setting the number of hypotheses beyond 64 did not improve results in either English or Chinese, so we reported runs with n-best set to 64. We believe this is because the two language-model based approaches make highly correlated ranking decisions based on character n-grams.

Held-out scores peaked with 5-grams for Chinese; 3-grams and 4-grams were not much worse and longer n-grams performed nearly identically. We used 7500 as the number of distinct characters, though this parameter is not at all sensitive to within an order of magnitude. We used LingPipe's default of setting the interpolation parameter equal to the n-gram length; for the final evaluation $\theta = 5.0$. Higher interpolation ratios favor precision over recall, lower ratios favor recall. Values within an order of magnitude performed with 1% F-measure and 2% precision/recall.

5 Bakeoff Time and Effort

The total time spent on this SIGHAN bakeoff was about 2 hours for the word segmentation task and 10 hours for the named-entity task (not including writing this paper). We started from a working word segmentation system for the last SIGHAN. Most of the time was spent munging entity data, with the rest devoted to held out analysis. The final code was roughly one page per task, with only a dozen or so LingPipe-specific lines. The final run, including unpacking, training and testing, took 45 minutes on a 512MB home PC; most of the time was named-entity decoding.

6 Results

Official bakeoff results for the four word segmentation corpora are shown in Figure 3, and for the two named entity corpora in Figure 4. Column labels are R for recall, P for precision, F_1 for balanced F-measure, *Best* F_1 for the best closed system's F_1 score, *OOV* for the out-of-vocabulary rate in the test corpus, and R_{OOV} for recall on the out-of-vocabulary items. For the named-entity results, precision and recall are also broken down by category.

7 Distribution

LingPipe may be downloaded from its homepage, `http://www.alias-i.com/lingpipe`. The code for the bakeoff is available via anonymous CVS from the sandbox. An Apache Ant makefile is provided to generate our bakeoff submission from the official data distribution format.

References

Carpenter, B. 2005. Scaling high-order character language models to gigabytes. *ACL Software Workshop*. Ann Arbor.

Culotta, A. and A. McCallum. 2004. Confidence estimation for information extraction. *HLT/NAACL 2004*. Boston.

Teahan, W. J., Y. Wen, R. McNab, and I. H. Witten. 2000. A compression-based algorithm for Chinese word segmentation. *Computational Linguistics*, **26**(3):375–393.