LING570 PROJECT 2

Jennifer Romelfanger Chris Curtis

December 4, 2012

In this project we evaluated combinations of morphological and structural features in morpheme-separated Korean text to train both MEMM and CRF models using the Mallet toolset. We achieved 96.25% morpheme accuracy and 22.11% sentence accuracy over the test data, which significantly improved on our prior results. We found that certain character-level features in particular were highly significant, reflecting probable regularities in Korean morphology.

1 DATA

For this problem we were provided with a training set consisting of 1475 Korean sentences drawn from the Morphologically Annotated Korean Text corpus[2]. The provided data consisted of romanized words separated into morphemes and tagged with a modified Treebank tagset. The data were divided into a training set of 1000 sentences and a test set of 475 sentences.

2 DESIGN

Our tagger was composed of four basic components: parser, feature extractor, classifier (provided by Mallet[3]), and front-end driver and evaluator. The tagger was built using insight gained from our previous Korean tagging homework [4, 1] but was developed as entirely new code.

2.1 *Parser*

An early version of the parser created a queue of morphemes with the feature extractor as a destination. Although there may be some advantages to this approach, we found that it was unwieldy and led to complex feature extraction logic that significantly hindered our ability to quickly test new feature ideas. This parser was discarded.

The final parser builds each sentence as a linked list of words, which are in turn built as linked lists of morphemes. This structure allows a great deal of flexibility in extracting features along with any relevant context.

2.2 *Feature Extractor*

The heart of our tagger is the feature extractor, which takes sentences parsed by the parser and emits a feature line for each morpheme to an

output file. The output file can be generated in a format suitable for either the MEMM or CRF (`SimpleTagger`) classifiers.

2.2.1 *Feature Definition and Control*

Features were defined entirely in code, which provided a great deal of flexibility in exploring the design of various features at the expense of run-time flexibility. Although each feature is controlled by a flag in our implementation, thus enabling automated comparison testing, a useful future enhancement would be to implement a domain-specific language for expressing feature templates in text parsed at run-time.[1]

2.2.2 *Feature Templates*

We identified several candidate feature templates through exploration and hands-on testing. These feature templates fell into five broad categories:

Parsing Control of how particular inputs are parsed

Morpheme Current morpheme, *n* previous/lookahead morphemes, joint morphemes

Word Current word, *n* previous/lookahead words

Structural Position of morpheme in sentence, in word, etc.

Orthographic Morpheme n-character prefix/suffix, bigrams, trigrams

Our conception of possible features was informed by our previous work on a simple Korean POS tagger, and superficial observation of Korean morphology and syntax. In particular, as an agglutinative language, we expected the role of morphemes to be relevant to the position in the word. We also expected the position of certain morphemes within a sentence (e.g. the first morpheme of the final word) to be highly relevant.

The individual feature templates we tested are given in Table 1.

2.3 *Classifier*

We tested our feature sets primarily using the Mallet MEMM classifier with Markov order 2 due to the rapid training/evaluation cycle. Once we identified an optimum feature set, we trained and tested a CRF model.

2.4 *Front-End*

Our front-end code manages the "administrative" aspects of the task, including control of the parser and feature extractor. It also performs evaluation against the gold standard tags to produce the required output file.

3 EVALUATION

3.1 *Feature Selection*

After the broad set of feature templates were identified and implemented, we performed some initial testing by hand to identify particularly strong

1 A dynamic feature-definition capability such as this would also lend itself to automated exploration of the potential feature space using e. g. genetic methods.

Features			
Class	Name	Value	Definition
Parsing	CollapseNumbers	token	`[\{Digit}]+` → `"<NUM>"`
Morpheme	MorphsHistory = n	morpheme(s)	$m_{-1}, \ldots, m_{-n}$
	MorphsFuture = n	morpheme(s)	$m_{+1}, \ldots, m_{+n}$
	MorphPrevPair	string	(m_{-1}, m_0)
	MorphPrevPrevPair	string	(m_{-2}, m_{-1})
	MorphTriple	string	(m_{-2}, m_{-1}, m_0)
	MorphMidTriple	string	(m_{-1}, m_0, m_{+1})
	MorphBack2	string	(m_{-2}, m_0)
	MorphForward2	string	(m_0, m_{+2})
Word	CurrWord	word	w_0
	WordHistory	word(s)	$w_0, \ldots, w_{-n}$
	WordFuture	word(s)	$w_0, \ldots, w_{+n}$
Structure	MorphEPosition	integer	$len(w) - pos_w(m)$
	MorphIsPenult	binary	$pos_w(m_0) = t-1$
	MorphIsPostPrefix	binary	$pos_w(m_0) = 2$
	MorphSPosition	$pos_w(m)$	$pos_w(m)$
	SentenceEPosition	integer	$len(s) - pos_s(w)$
	SentencePosFirstOnly	control	use pos_s only for m_1 in word
	SentenceSPosition	integer	$pos_s(w)$
	WordEnd	binary	$pos_w(m_0) = t$
	WordPenult	string	m_{t-1}
	WordStart	binary	$pos_w(m_0) = 1$
Orthographic	CharBigrams	string	char 2-grams
	CharTrigrams	strings	char 3-grams
	CharPrefix{1,2,3}	string	{1,2,3}-prefix
	CharSuffix{1,2,3}	string	{1,2,3}-suffix

Table 1: Feature templates

Features					Accuracy		
hist[1]	future[2]	splits[3]	3-prefix[4]	trigrams[5]	Morpheme	Sentence	Unseen
1	0	yes	no	no	**96.7793**	32.6737	86.4686
1	0	yes	yes	no	96.7400	**33.6634**	87.1287
2	1	no	yes	yes	96.7596	29.7030	**87.4587**

[1] MorphHist
[2] MorphFuture
[3] Both MorphBack2 and MorphForward2
[4] CharPrefix3
[5] CharTrigrams

Table 2: Devtest Results

or weak features. This first pass showed that the word-level features were useless at best, and often reduced the accuracy by 10% or more, so they were not included in further evaluations. We also chose to always replace numbers with a single token (*CollapseNumbers*), as there seemed to be no theoretical basis for treating individual numbers as distinct morphemes/words.[2]

The next step in our analysis was automated large-scale testing of combinations of features, against 10% (n=101) held-out training sentences. We tested 281 different combinations of features, achieving maximum accuracies of 96.78% by morpheme, 33.66% by sentence, and 87.46% for unseen morphemes. Although different combinations of features maximized each metric, the per-morpheme accuracy only varied by 0.03% regardless of the metric optimized for. The devtest results are summarized in Table 2.

3.2 *Feature Evaluation*

We then evaluated these feature combinations against the test data. Although the accuracies were, as expected, lower, the results were still within 1% of the devtest performance. In addition, a single feature set gave the best results on all three metrics. These results are summarized in Table 3.

3.3 *Feature Space Exploration*

To further evaluate the performance of various features, we then applied our automated testing to the test data. Although we did not exhaustively search the combinatorial feature space, we did test 1022 different combi-

2 This assumes that Korean is not marked based on number *value*. In a hypothetical language where 100 and 10 have different distributions, this assumption would obviously not hold. In the absence of any relevant training data we also ignored the potential impact of differing treatment of noun classes.

Features					Accuracy		
hist	future	splits	3-prefix	trigrams	Morpheme	Sentence	Unseen
1	0	yes	no	no	95.7776	19.5789	85.9387
1	0	yes	yes	no	95.7863	19.1579	85.9760
2	1	no	yes	yes	**96.2463**	**22.1053**	**86.9110**

Table 3: Test Results

nations. The highest-performing model in these tests achieved 96.2463% morpheme accuracy, 20.6316% sentence accuracy, and 87.472% unseen-morpheme accuracy. This model was nearly identical to our devtest-selected model, but with an additional morpheme of lookahead (*MorphFuture*=2). Although this enhanced the unseen-morpheme accuracy by approximately 0.5%, it also decreased our sentence accuracy by nearly 2%.

3.4 *Statistical Meta-analysis*

We then performed regression analysis[3] on the feature combinations data in order to identify which features contributed most significantly to accuracy. The first analysis treated all CharSuffix and CharPrefix combinations as a single feature, i.e. either all 1-, 2-, and 3-character prefixes and suffixes, or none; this feature was consistently the strongest contributor to accuracy ($\hat{\beta} = 0.5154$, $p < 0.001$). Other strong predictors ($p < 0.001$) were CharBigrams, CharTrigrams, MorphHist, WordStart, and MorphSPos. These results are summarized in Table 4.

We then performed a narrower analysis to isolate the predictive power of the character prefix and suffix features by holding all other features constant and only varying the mixture of character suffixes and prefixes. The most significant factor here for morpheme accuracy was CharTrigrams ($\hat{\beta} = 0.5747$, $p < 0.001$); CharBigrams was the strongest predictor for sentence ($\hat{\beta} = 4.069$, $p < 0.001$) and unseen morpheme ($\hat{\beta} = 1.171$, $p < 0.001$) accuracy. These results are summarized in Table 5.

3.5 *CRF Classifier*

Finally, we attempted to train CRF models using our top feature set. With the same features enabled and Markov order 1, the Mallet CRF classifier achieved morpheme accuracy of 94.1026%, sentence accuracy of 12.6316%, and accuracy on unseen morphemes of 84.1810%. We attempted to train CRF models with Markov order 2, but were unsuccessful in running a training cycle to completion.[4]

3 Thanks to Micah Jensen (Georgetown University) for his assistance with statistical interpretation. Errors in analysis are solely ours, not his.

4 The average training run was in progress for well over 24 hours before appearing to stall.

	$\hat{\beta}$		
Feature	Morpheme	Sentence	Unseen
CharSuffix	0.5154	0.5030	0.6424
CharBigrams	0.3689	0.4602	0.4408
CharTrigrams	0.3223	0.3060	0.2541
MorphHist	0.2249	0.3170	0.2079
WordStart	0.1673	0.1230	0.0924
MorphSPos	0.1028	0.0848	0.0576

Table 4: Feature Statistics (all $p < 0.001$)

	$\hat{\beta}$		
Feature	Morpheme	Sentence	Unseen
CharSuffix1	0.09911^{2}	0.5105^{1}	0.8612^{3}
CharSuffix2	0.1648^{3}	NS	0.9437^{3}
CharSuffix3	0.3606^{3}	1.686^{3}	1.923^{3}
CharPrefix1	0.07948^{1}	0.6343^{2}	0.8096^{3}
CharPrefix2	0.2219^{3}	0.6962^{2}	NS
CharPrefix3	NS	NS	NS
CharBigrams	0.5569^{3}	4.069^{3}	1.171^{3}
CharTrigrams	0.5747^{3}	1.160^{3}	0.8302^{3}

[1] $p < 0.05$
[2] $p < 0.01$
[3] $p < 0.001$
[NS] Not significant

Table 5: Character-Level Feature Statistics

4 DISCUSSION

4.1 *Features*

As Korean is an agglutinating language, we expected the position of morphemes within the word and the sentence to be significant, and this was indeed the case in our testing. We also expected the relative positions of morphemes to be significant, reflecting the SOV word order and presumed rules for combining affixes. This was also borne out by our testing, although to a much lesser degree than anticipated. We suppose this may be partially due to the characteristics of this corpus; as all sentences were news articles from a single agency, the range of usage and variation in structure and register may be significantly constraint as compared to a larger corpus.

The features that provided the most incremental gain, however, were the character-level features. On average, these features increased morpheme accuracy by a full 3%. We suspect this unexpectedly-large effect may be due to fusional and/or morphophonemic (e.g. *-eul*/*-reul*) aspects of Korean that we do not have access to directly.

4.2 *Classifiers*

Our biggest disappointment in this project was the difficulty in working with Mallet's CRF classifier. Once we had identified a good candidate feature set using the MEMM model, the Markov order 1 CRF model was significantly less accurate. This is not especially surprising given that the MEMM model was order 2, but the massive increase in training time (not to mention failure to complete) made any potential accuracy improvement largely irrelevant.

5 SUMMARY AND FUTURE WORK

The project was a mixture of great successes (3% improvement with a handful of features) and great disappointments (the CRF classifier). It was particularly gratifying to make such dramatic improvements over our previous attempts: 91.7% [4] and 90.8% [1]. It was also interesting to find

that while significant improvement was due to applying a trigram model, good feature selection had an even more significant impact.

The logical next step would be to evaluate these models against an expanded corpus, again seeking to identify the most effective features. Another major next step would be training a higher-performance CRF implementation; the difficulties we faced made it impossible to make any meaningful comparisons to the MEMM implementation.

REFERENCES

[1] Chris Curtis. LING570 Homework 3 Report. 2012.

[2] Na-Rae Han. Morphologically annotated korean text. `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2004T03`, 2004.

[3] Andrew Kachites McCallum. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu, 2002.

[4] Jennifer Romelfanger. LING570 Homework 3 Report. 2012.