

Robust *kaomoji* detection in Twitter

Steven Bedrick, Russell Beckley, Brian Roark, Richard Sproat

Center for Spoken Language Understanding, Oregon Health & Science University
Portland, Oregon, USA

Abstract

In this paper, we look at the problem of robust detection of a very productive class of Asian style emoticons, known as facemarks or *kaomoji*. We demonstrate the frequency and productivity of these sequences in social media such as Twitter. Previous approaches to detection and analysis of *kaomoji* have placed limits on the range of phenomena that could be detected with their method, and have looked at largely monolingual evaluation sets (e.g., Japanese blogs). We find that these emoticons occur broadly in many languages, hence our approach is language agnostic. Rather than relying on regular expressions over a predefined set of likely tokens, we build weighted context-free grammars that reward graphical affinity and symmetry within whatever symbols are used to construct the emoticon.

1 Introduction

Informal text genres, such as email, SMS or social media messages, lack some of the modes used in spoken language to communicate affect – prosody or laughter, for example. Affect can be provided within such genres through the use of text formatting (e.g., capitalization for emphasis) or through the use of extra-linguistic sequences such as the widely used smiling, winking ;) emoticon. These sorts of vertical face representations via ASCII punctuation sequences are widely used in European languages, but in Asian informal text genres another class of emoticons is popular, involving a broader symbol set and with a horizontal facial orientation. These go by the name of facemarks or *kaomoji*. Figure 1 presents

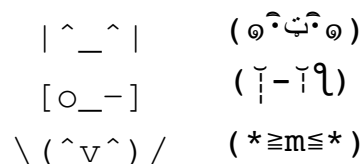


Figure 1: Some representative *kaomoji* emoticons

several examples of these sequences, including both relatively common *kaomoji* as well as more exotic and complex creations.

This class of emoticon is far more varied and productive than the sideways European style emoticons, and even lists of on the order of ten thousand emoticons will fail to cover all instances in even a modest sized sample of text. This relative productivity is due to several factors, including the horizontal orientation, which allows for more flexibility in configuring features both within the face and surrounding the face (e.g., arms) than the vertical orientation. Another important factor underlying *kaomoji* productivity is historical in nature. *kaomoji* were developed and popularized in Japan and other Asian countries whose scripts have always required multibyte character encodings, and whose users of electronic communication systems have significant experience working with characters beyond those found in the standard ASCII set.

Linguistic symbols from various scripts can be appropriated into the *kaomoji* for their resemblance to facial features, such as a winking eye, and authors of *kaomoji* sometimes use advanced Unicode techniques to decorate glyphs with elaborate combinations of diacritic marks. For example, the *kaomoji*

moji in the top righthand corner of Figure 1, includes an Arabic letter, and Thai vowel diacritics. Accurate detection of these tokens – and other common sequences of extra-linguistic symbol sequences – is important for normalization of social media text for downstream applications.

At the most basic level, the complex and unpredictable combinations of characters found within many *kaomoji* (often including punctuation and whitespace, as well as irregularly-used Unicode combining characters) can seriously confound sentence and word segmentation algorithms that attempt to operate on *kaomoji*-rich text; since segmentation is typically the first step in any text processing pipeline, issues here can cause a wide variety of problems downstream. Accurately removing or normalizing such sequences before attempting segmentation can ensure that existing NLP tools are able to effectively work with and analyze *kaomoji*-including text.

At a higher level, the inclusion of a particular *kaomoji* in a text represents a conscious decision on the part of the text’s author, and fully interpreting the text necessarily involves a degree of interpretation of the *kaomoji* that they chose to include. European-style emoticons form a relatively closed set and are often fairly straightforward to interpret (both in terms of computational, as well as human, effort); *kaomoji*, on the other hand, are far more diverse, and interpretation is rarely simple.

In this paper, we present preliminary work on defining robust models for detecting *kaomoji* in social media text. Prior work on detecting and classifying these extra-linguistic sequences has relied on the presence of fixed attested patterns (see discussion in Section 2) for detection, and regular expressions for segmentation. While such approaches can capture the most common *kaomoji* and simple variants of them, the productive and creative nature of the phenomenon results in a non-negligible out-of-vocabulary problem. In this paper, we approach the problem by examining a broader class of possible sequences (see Section 4.2) for symmetry using a robust probabilistic context-free grammar with rule probabilities proportional to the symmetry or affinity of matched terminal items in the rule. Our PCFG is robust in the sense that every candidate sequence is guaranteed to have a valid parse. We use the re-

sulting Viterbi best parse to provide a score to the candidate sequence – reranking our high recall list to achieve, via thresholds, high precision. In addition, we investigate unsupervised model adaptation, by incorporating Viterbi-best parses from a small set of attested *kaomoji* scraped from websites; and inducing grammars with a larger non-terminal set corresponding to regions of the face.

We present bootstrapping experiments for deriving highly functional, language independent models for detecting *kaomoji* in text, on multilingual Twitter data. Our approach can be used as part of a stand-alone detection model, or as input into semi-automatic *kaomoji* lexicon development. Before describing our approach, we will first present prior work on this class of emoticon.

2 Prior Work

Nakamura et al. (2003) presented a natural language dialogue system that learned a model for generating *kaomoji* face marks within Japanese chat. They trained a neural net to produce parts of the emoticon – mouth, eyes, arms and “optional things” as observed in real world data. They relied on a hand-constructed inventory of observed parts within each of the above classes, and stitched together predicted parts into a complete *kaomoji* using simple templates.

Tanaka et al. (2005) presented a finite-state chunking approach for detecting *kaomoji* in Japanese on-line bulletin boards using SVMs with simple features derived from a 7 character window. Training was performed on *kaomoji* dictionaries found online. They achieved precision and recall in the mid-80s on their test set, which was a significant recall improvement (17% absolute) and modest precision improvement (1.5%) over exact match within the dictionaries. They note certain kinds of errors, e.g., “(Thu)” which demonstrate that their chunking models are (unsurprisingly) not capturing the typical symmetry of *kaomoji*. In addition, they perform classification of the *kaomoji* into 6 rough categories (happy, sad, angry, etc.), achieving high performance (90% accuracy) using a string kernel within an SVM classifier.

Ptaszynski et al. (2010) present work on a large database of *kaomoji*, which makes use of an analy-

sis of the gestures conveyed by the emoticons and their relation to a theory of non-verbal expressions. They created an extensive (approximately 10,000 entry) lexicon with 10 emotion classes, and used this database as the basis of both emoticon extraction from text and emotion classification. To detect an emoticon in text, their system (named ‘CAO’) looked for three symbols in a row from a vocabulary of the 455 most frequent symbols in their database. Their approach led to a 2.4% false negative rate when evaluated on 1,000 sentences extracted from Japanese blogs. Once detected, the system extracts the emoticon from the string using a gradual relaxation from exact match to approximate match, with various regular expressions depending on specific partial match criteria. A similar deterministic algorithm based on sequenced relaxation from exact match was used to assign affect to the emoticon.

Our work focuses on the emoticon detection stage, and differs from the above systems in a number of ways. First, while *kaomiji* were popularized in Asia, and are most prevalent in Asian languages, they not only found in messages in those languages. In Twitter, which is massively multilingual, we find *kaomiji* with some frequency in many languages, including European languages such as English and Portuguese, Semitic languages and a range of Asian languages. Our intent is to have a language independent algorithm that looks for such sequences in any message. Further, while we make use of online dictionaries as development data, we appreciate the productivity of the phenomenon and do not want to restrict the emoticons that we detect to those consisting of pre-observed characters. Hence we focus instead on characteristics of *kaomiji* that have been ignored in the above models: the frequent symmetry of the strings. We make use of context-free models, built in such a way as to guarantee a parse for any candidate sequence, which permits exploration of a much broader space of potential candidates than the prior approaches, using very general models and limited assumptions about the key components of the emoticons.

3 Data

Our starting resources consisted of a large, multilingual corpus of Twitter data as well as a smaller

collection of *kaomiji* scraped from Internet sources. Our Twitter corpus consists of approximately 80 million messages collected using Twitter’s “Streaming API” over a 50-day period from June through August 2011. The corpus is extremely linguistically diverse; human review of a small sample identified messages written in >30 languages. The messages themselves exhibit a wide variety of phenomena, including substantial use of different types of Internet slang and written dialect, as well as numerous forms of non-linguistic content such as emoticons and “ASCII art.”

We took a two-pronged approach to developing a set of “gold-standard” *kaomiji*. Our first approach involved manually “scraping” real-world examples from the Internet. Using a series of handwritten scripts, we harvested 9,193 examples from several human-curated Internet websites devoted to collecting and exhibiting *kaomiji*. Many of these consisted of several discrete sub-units, typically including at least one “face” element along with a small amount of additional content. For example, consider the following *kaomiji*, which appeared in this exact form eight times in our Twitter corpus: ヽ(❁ゝ)ノおはよお〜♥. Note that, in this case, the “face” is followed by a small amount of hiragana, and that the message concludes with a dingbat in the form of a “heart” symbol.¹

Of these 9,193 scraped examples, we observed $\approx 3,700$ to appear at least once in our corpus of Twitter messages, and $\approx 2,500$ more than twice. The most common *kaomiji* occurred with frequencies in the low hundreds of thousands, although the frequency with which individual *kaomiji* appeared roughly followed a power-law distribution, meaning that there were a small number that occurred with great frequency and a much larger number that only appeared rarely.

From this scraped corpus, we attempted to identify a subset that consisted solely of “faces” to serve as a high-precision training set. After observing that nearly all of the faces involved a small number of characters bracketed one of a small set of natural grouping characters (parentheses, “curly braces,”

¹Note as well that this *kaomiji* includes not only a wide variety of symbols, but that some of those symbols are themselves modified using combining diacritic marks. This is a common practice in modern *kaomiji*, and one that complicates analysis.

etc.), we extracted approximately 6,000 substrings matching a very simple regular expression pattern. This approach missed many *kaomoji*, and of the examples that it did detect, many were incomplete (in that they were missing any extra-bracketed content— arms, ears, whiskers, etc.) However, the contents of this “just faces” sub-corpus offered decent coverage of many of the core *kaomoji* phenomena in a relatively noise-free manner. As such, we found it to be useful as “seed” data for the grammar adaptation described in section 4.4.

In addition to our “scraped” *kaomoji* corpus, we constructed a smaller corpus of examples drawn directly from our Twitter corpus. The *kaomoji* phenomenon is complex enough that capturing it in its totality is difficult. However, it is possible to capture a subset of *kaomoji* by looking for regions of perfect lexical symmetry. This approach will capture many of the more regularly-formed and simple *kaomoji* (for example, $\hat{\ }(-_-\)\hat{\ }$), although it will miss many valid *kaomoji*. Using this approach, we identified 3,580 symmetrical candidate sequences; most of these were indeed *kaomoji*, although there were several false positives (for example, symmetrical sequences of repeated periods, question marks, etc.). Using simple regular expressions, we were able to remove 289 such false positives.

Interestingly, there was very little overlap between the corpus scraped from the Web and the symmetry corpus. A total of 39 *kaomoji* appeared in exactly the same form in both sets. We noted, however, that the *kaomoji* harvested from the Web tended to be longer and more elaborate than those identified from our Twitter corpus using the symmetry heuristic (Mann-Whitney U, $p < 0.001$), and as previously discussed, the Web *kaomoji* often contained one or more face elements. Thus we expanded our definition of overlap, and counted sequences from the symmetrical corpus that were substrings of scraped *kaomoji*. Using this criterion, we identified 177 possibly intersecting *kaomoji*. The fact that so few individual examples occurred in both corpora illustrates the extremely productive nature of the phenomenon.

4 Methods

4.1 Graphical similarity

The use of particular characters in *kaomoji* is ultimately based on their graphical appearance. For

0.9500	208d	fe5a	()
0.9500	207d	fe5a	()
0.9500	1fef	2034	`	”
0.9500	1fef	2033	`	”
0.9500	1fef	2032	`	”
0.9500	1489	1494	∩	∩
0.9500	055b	1fef	`	`
0.9500	0500	13cf	d	b
0.9500	0351	2e12	‘	,
0.9500	013f	14b2	Б	∩

Figure 2: Ten example character pairs with imperfect (but very high) symmetry identified by our algorithm. Columns are: score, hex code point 1, hex code point 2, glyph 1, glyph 2.

example, good face delimiters frequently include mated brackets or parentheses, since these elements naturally look as if they delimit material. Furthermore, there are many characters which are not technically “paired,” but look roughly more-or-less symmetrical. For example, the Arabic-Indic digits $\mathfrak{9}$ and $\mathfrak{6}$ are commonly used as bracketing delimiters, for example: $\mathfrak{9}^{\circ}\mathfrak{6}^{\circ}$. These characters can serve both as “arms” as well as “ears.”

Besides bracketing, symmetry plays an additional role in *kaomoji* construction. Glyphs that make good “eyes” are often round; “noses” are often symmetric about their central axis. Therefore a measure of graphical similarity between characters is desirable.

To that end, we developed a very simple measure of similarity. From online sources, we downloaded a sample glyph for each code point in the Unicode Basic Multilingual Plane, and extracted a bitmap for each. In comparing two glyphs we first scale them to have the same aspect ratio if necessary, and we then compute the proportion of shared pixels between them, with a perfect match being 1 and the worst match being 0. We can thus compute whether two glyphs look similar; whether one glyph is a good mirror image of the other (by comparing glyph A with the mirror image of glyph B); and whether a glyph is (vertically) symmetric (by computing the similarity of the glyph and its vertical mirror image).

The method, while clearly simple-minded, nonetheless produces plausible results, as seen in Figure 2, which shows the best 10 candidates for mirror image character pairs. We also calculate the same score without flipping the image vertically, which is also used to score possible symbol matches, as detailed in Section 4.3.

4.2 Candidate extraction

We perform candidate *kaomoji* extraction via a very simple hidden Markov model, which segments all strings of Unicode graphemes into contiguous regions that are either primarily linguistic (mainly language encoding symbols²) or primarily non-linguistic (mainly punctuation, or other symbols). Our candidate emoticons, then, are this extensive list of mainly non-linguistic symbol sequences. This is a high recall approach, returning most sequences that contain valid emoticons, but quite low precision, since it includes many other sequences as well (extended runs of punctuation, etc.).

The simple HMM consists of 2 states: call them *A* (mainly linguistic) and *@* (mainly non-linguistic). Since there are two emitted symbol classes (linguistic *L* and non-linguistic *N*), each HMM state must have two emission probabilities, one for its dominant symbol class (*L* in *A* and *N* in *@*) and one for the other symbol class. Non-linguistic symbols occur quite often in linguistic sequences, as punctuation for example. However, sequences of, say, 3 or more in a row are not particularly frequent. Similarly, linguistic symbols occur often in *kaomoji*, though not often in sequences of, say, 3 or more. Hence, to segment into contiguous sequences of a certain number in a row, the probability of transition from state *A* to state *@* or vice versa must be significantly lower than the probability of emitting one or two *N* from *A* states or *L* from *@* states. We thus have an 8 parameter HMM (four transition and four emission probabilities) that was coarsely parameterized to have the above properties, and used it to extract candidate non-linguistic sequences for evaluation by our PCFG model.

Note that this approach does have the limitation that it will trim off some linguistic symbols that occur on the periphery of an emoticon. Future versions of this part of the system will address this issue by extending the HMM. For this paper, we made use of a slightly modified version of this simple HMM for candidate extraction. The modifications involved the addition of a special input state for whitespace and full-stop punctuation, which helped prevent certain very common classes of false-positive.

²Defined as a character having the Unicode “letter” character property.

rule	score	rule	score
$X \rightarrow a X b$	$S(a,b)$	$X \rightarrow a b$	$S(a,b)$
$X \rightarrow a X$	ϵ	$X \rightarrow X a$	ϵ
$X \rightarrow a$	δ	$X \rightarrow X X$	γ

Table 1: Rule schemata for producing PCFG

4.3 Baseline grammar induction

We perform a separate PCFG induction for every candidate emoticon sequence, based on a small set of rule templates methods for assigning rule weights. By inducing small, example-specific PCFGs, we ensure that every example has a valid parse, without growing the grammar to the point that the grammar constant would seriously impact parser efficiency.

Table 1 shows the rule schemata that we used for this paper. The resulting PCFG would have a single non-terminal (*X*) and the variables *a* and *b* would be instantiated with terminal items taken from the candidate sequence. Each instantiated rule receives a probability proportional to the assigned score. For the rules that “pair” symbols *a* and *b*, a score is assigned in two ways, call them $S_1(a, b)$ and $S_2(a, b)$ (they will be defined in a moment). Then $S(a, b) = \max(S_1(a, b) \text{ and } S_2(a, b))$. If $S(a, b) < \theta$, for some threshold θ ,³ then no rule is generated. S_1 is the graphical similarity of the first symbol with the vertical mirror image of the second symbol, calculated as presented in Section 4.1. This will give a high score for things like balanced parentheses. S_2 is the graphical similarity of the first symbol with the second symbol (not vertically flipped), which gives high scores to the same or similar symbols. This permits matches for, say, eyes that are not symmetric due to an orientation of the face, e.g., ([^]↷). The other parameters (ϵ , δ and γ) are included to allow for, but penalize, unmatched symbols in the sequence.

All possible rules for a given sequence are instantiated using these templates, by placing each symbol in the *a* slot with all subsequent symbols in the *b* slot and scoring, as well as creating all rules with just *a* alone for that symbol. For example, if we are given the *kaomoji* ($\circ_o;)$ specific rules would be created if the similarity scores were above threshold. For the second symbol ‘*o*’, the algorithm would evaluate the

³For this paper, θ was chosen to be 0.7.

similarity between ‘o’ and each of the four symbols to its right – , o, ; and) .

The resulting PCFG is normalized by summing the score for each rule and normalizing by the score. The grammar is then transformed to a weakly equivalent CNF by binarizing the ternary rules and introducing preterminal non-terminals. This grammar is then provided to the parser⁴, which returns the Viterbi best parse of the candidate emoticon along with its probability. The score is then converted to an approximate perplexity by dividing the negative log probability by the number of unique symbols in the sequence and taking the exponential.

4.4 Grammar enhancement and adaptation

The baseline grammar induction approach outlined in the previous section can be improved in a couple of ways, without sacrificing the robustness of the approach. One way is through grammar adaptation based on automatic parses of attested *kaomoji*. The other is by increasing the number of non-terminals in the grammar, according to a prior understanding of their typical (canonical) structure. We shall discuss each in turn.

Given a small corpus of attested emoticons (in our case, the “just faces” sub-corpus described in section 3), we can apply the parser above to those examples, and extract the Viterbi best parses into an automatically created treebank. From that treebank, we extract counts of rule productions and use these rule counts to inform our grammar estimation. The benefit of this approach is that we will obtain additional probability mass for frequently observed constructions in that corpus, thus preferring commonly associated pairs within the grammar. Of course, the corpus only has a small fraction of the possible symbols that we hope to cover in our robust approach, so we want to incorporate this information in a way that does not limit the kinds of sequences we can parse.

We can accomplish this by using simple Maximum a Posteriori (MAP) adaptation of the grammar (Bacchiani et al., 2006). In this scenario, we will first use our baseline method of grammar induction, using the schemata shown in Table 1. The scores derived in that process then serve as prior counts

⁴We used the BUBS parser (Bodenstab et al., 2011). <http://code.google.com/p/bubs-parser/>

for the rules in the grammar, ensuring that all of these rules continue to receive probability mass. We then add in the counts for each of the rules from the treebank. Many of the rules may have been unobserved in the corpus, in which case they receive no additional counts; observed rules, however, will receive extra weight proportional to their frequency in that corpus. Note that these additional weights can be scaled according to a given parameter. After incorporating these additional counts, the grammar is normalized and parsing is performed as before. Of course, this process can be iterated – a new automatic treebank can be produced based on an adapted grammar, and so on.

In addition to grammar adaptation, we can enrich our grammars by increasing the non-terminal sets. To do this, we created a nested hierarchy of “regions” of the emoticons, with constraints related to the canonical composition of the faces, e.g., eyes are inside of faces, noses/mouths between eyes, etc. These non-terminals replace our generic non-terminal *X* in the rule schemata. For the current paper, we included the following five “region” non-terminals: *ITEM*, *OUT*, *FACE*, *EYES*, *NM*. The non-terminal *ITEM* is intended as a top-most non-terminal to allow multiple emoticons in a single sequence, via an $ITEM \rightarrow ITEM\ ITEM$ production. None of the others non-terminals have repeating productions of that sort – so this replaces the $X \rightarrow X\ X$ production from Table 1.

Every production (other than $ITEM \rightarrow ITEM\ ITEM$) has zero or one non-terminals on the right-hand side. In our new schemata, non-terminals on the left-hand side can only have non-terminals on the right-hand side at the same or lower levels. This enforces the nesting constraint, i.e., that eyes are inside of the face. Levels can be omitted however – e.g., eyes but no explicit face delimiter – hence we can “skip” a level using unary projections, e.g., $FACE \rightarrow EYES$. Those will come with a “skip level” weight. Categories can also rewrite to the same level (with a “stay level” weight) or rewrite to the next level after emitting symbols (with a “move to next level” weight).

To encode a preference to move to the next level rather than to stay at the same level, we assign a weight of 1 to moving to the next level and a weight of 0.5 to staying at the same level. The “skip”

rule		score
ITEM	→ ITEM ITEM	γ
ITEM	→ OUT	γ
OUT	→ a OUT b	$S(a,b) + 0.5$
OUT	→ a OUT	$\epsilon + 0.5$
OUT	→ OUT a	$\epsilon + 0.5$
OUT	→ a FACE b	$S(a,b) + 1$
OUT	→ a FACE	$\epsilon + 1$
OUT	→ FACE a	$\epsilon + 1$
OUT	→ FACE	0.5
FACE	→ a FACE b	$S(a,b) + 0.5$
FACE	→ a FACE	$\epsilon + 0.5$
FACE	→ FACE a	$\epsilon + 0.5$
FACE	→ a EYES b	$S(a,b) + 1$
FACE	→ a EYES	$\epsilon + 1$
FACE	→ EYES a	$\epsilon + 1$
FACE	→ EYES	0.1
EYES	→ a EYES b	$S(a,b) + 0.5$
EYES	→ a EYES	$\epsilon + 0.5$
EYES	→ EYES a	$\epsilon + 0.5$
EYES	→ a NM b	$S(a,b) + 1$
EYES	→ a NM	$\epsilon + 1$
EYES	→ NM a	$\epsilon + 1$
EYES	→ NM	0.1
EYES	→ a b	$S(a,b) + 1$
NM	→ a NM	ϵ
NM	→ NM a	ϵ
NM	→ a	δ

Table 2: Rule schemata for expanded non-terminal set

weights depend on the level, e.g., skipping OUT should be cheap (weight of 0.5), while skipping the others more expensive (weight of 0.1). These weights are like counts, and are added to the similarity counts when deriving the probability of the rule. Finally, there is a rule in the schemata in Table 1 with a pair of symbols and no middle non-terminal. This is most appropriate for eyes, hence will only be generated at that level. Similarly, the single symbol on the right-hand side is for the NM (nose/mouth) region. Table 2 presents our expanded rule schemata.

Note that the grammar generated with this expanded set of non-terminals is robust, just as the earlier grammar is, in that every sequence is guaranteed to have a parse. Further, it can be adapted using the same methods presented earlier in this section.

5 Experimental Results

Using the candidate extraction methodology described in section 4.2, we extracted 1.6 million distinct candidates from our corpus of 80 million Twitter messages (candidates often appeared in multiple messages). These candidates included genuine emoticons, as well as extended strings of punctuation and other “noisy” chunks of text. Genuine *kaomoji* were often picked up with some amount of leading or trailing punctuation, for example: “. \(\` \nabla \`)/”; other times, *kaomoji* beginning with linguistic characters were truncated: $(\wedge, *)\int$.

We provided these candidates to our parser under four different conditions, each one producing 1.5 million parse trees: the single non-terminal approach described in section 4.3 or the enhanced multiple non-terminal approach described in section 4.4, both with and without training via the Maximum A Posteriori approach described in section 4.4.

Using the weighted-inside-score method described in section 4.3, we produced a ranked list of candidate emoticons from each condition’s output. “Well-scoring” candidates were ones for which the parser was able to construct a low-cost parse. We evaluated our approach in two ways. The first way examined precision—how many of the best-scoring candidate sequences actually contained *kaomoji*? Manually reviewing all 1.6 million candidates was not feasible, so we evaluated this aspect of our system’s performance on a small subset of its output. Computational considerations forced us to process our large corpus in parallel, meaning that our set of 1.6 million candidate *kaomoji* was already partitioned into 160 sets of $\approx 10,000$ candidates each. We manually reviewed the top 1,000 sorted results from one of these partitions, and flagged any entries that did not contain or consist of a face-like *kaomoji*. The results of each condition are presented in table 3.

The second evaluation approach we will examine looks at how our method compares with the trigram-based approach described by (Yamada et al., 2007) (as described by (Ptaszynski et al., 2010)). We trained both smoothed and unsmoothed language models⁵ on the “just faces” sub-corpus used for the A Posteriori grammar enhancement, and computed perplexity measurements for the same set $\approx 10,000$ candidates used previously. Table 3 presents these results; clearly, a smoothed trigram model can achieve good results. The unsmoothed model at first glance seems to have performed very well; note, however, that only approximately 600 (out of nearly 10,000) candidates were “matched” by the unsmoothed model (i.e., they did not contain any OOV symbols and therefore had finite perplexity scores), yielding a very small but high-precision set of emoticons.

Looking at precision, the model-based approaches outperformed our grammar approach. It

⁵Using the OpenGrm ngram language modeling toolkit.

Condition	P@1000	MAP
Single Nonterm, Untrained	0.662	0.605
Single Nonterm, Trained	0.80	0.945
Multiple Nonterm, Untrained	0.795	0.932
Multiple Nonterm, Trained	0.885	0.875
Unsmoothed 3-gram	0.888	0.985
Smoothed 3-gram	0.905	0.956
Mixed, Single Nonterm, Untrained	0.662	0.902
Mixed, Single Nonterm, Trained	0.804	0.984
Mixed, Multiple Nonterm, Untrained	0.789	0.932
Mixed, Multiple Nonterm, Trained	0.878	0.977

Table 3: Experimental Results.

should be noted, however, that the trigram approach was much less tolerant of certain non-standard formulations involving novel characters or irregular formulations (◀☺) and /◀◊) are examples of *kaomoji* that our grammar-based approach ranked more highly than did the trigram approach). The two approaches also had different failure profiles. The grammar approach’s false positives tended to be symmetrical sequences of punctuation, whereas the language models’ were more variable. Were we to review a larger selection of candidates, we believe that the structure-capturing nature of the grammar approach would enable it to outperform the more simplistic approach.

We also attempted a hybrid “mixed” approach in which we used the language models to re-rank the top 1,000 “best” candidates from our parser’s output. This generally resulted in improved performance, and for some conditions the improvement was substantial. Future work will explore this approach in greater detail and over larger amounts of data.

6 Discussion

We describe an almost entirely unsupervised approach to detecting *kaomoji* in irregular, real-world text. In its baseline state, our system is able to accurately identify a large number of examples using a very simple set of templates, and can distinguish *kaomoji* from other non-linguistic content (punctuation, etc.). Using minimal supervision, we were able to effect a dramatic increase in our system’s performance. Visual comparison of the “untrained” results with the “trained” results was instructive. The untrained systems’ results were very heavily influenced by their template rules’ strong preference for visual symmetry. Many instances of symmetrical punctuation sequences (e.g., . . ? . .) ended up being ranked more highly than even fairly sim-

ple *kaomoji*, and in the absence of other information, the length of the input strings also played a too-important role in their rankings.

The MAP-enhanced systems’ results, on the other hand, retained their strong preference for symmetry, but were also influenced by the patterns and characters present in their training data. For example, two of the top-ranked “false positives” from the enhanced system were the sequences >, < and = =, both of which (while symmetrical) also contain characters often seen in *kaomoji*. By using more structurally diverse training data, we expect further improvements in this area. Also, our system currently relies on a very small number of relatively simplistic grammar templates; expanding these to encode additional structure may also help.

Due to our current scoring mechanism, our parser is biased against certain categories of *kaomoji*. Particularly poorly-scored are complex creations such as (((◀◊◊◊◊◊))). In this example, the large number of combining characters and lack of obvious nesting therein confounded our templates and produced expensive parse trees. Future work will involve improved handling of such cases, either by modified parsing schemes or additional templates.

One other area of future work is to match particular *kaomoji*, or fragments of *kaomoji* (e.g. particular eyes), to particular affective states, or other features of the text. Some motifs are already well known: for example, there is wide use of TT, or the similar-looking Korean *hangeul* vowel *yu*, to represent crying eyes. We propose to do this initially by computing the association between particular *kaomoji* and words in the text. Such associations may yield more than just information on the likely affect associated with a *kaomoji*. So, for example, using pointwise mutual information as a measure of association, we found that in our Twitter corpus, (*^_*^*) seems to be highly associated with tweets about Korean pop music, *-_* with Brazilian postings, and °^(^v^)^° with Indonesian postings. Such associations presumably reflect cultural preferences, and could prove useful in identifying the provenance of a message even if more conventional linguistic techniques fail.

References

- Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.
- Nathan Bodenstab, Aaron Dunlop, Keith Hall, and Brian Roark. 2011. Adaptive beam-width prediction for efficient cyk parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 440–449.
- Junpei Nakamura, Takeshi Ikeda, Nobuo Inui, and Yoshiyuki Kotani. 2003. Learning face mark for natural language dialogue system. In *Proc. Conf. IEEE Int’l Conf. Natural Language Processing and Knowledge Eng*, pages 180–185.
- Michał Ptaszynski, Jacek Maciejewski, Paweł Dybala, Rafał Rzepka, and Kenji Araki. 2010. Cao: A fully automatic emoticon analysis system based on theory of kinesics. *IEEE Transactions on Affective Computing*, 1:46–59.
- Yuki Tanaka, Hiroya Takamura, and Manabu Okumura. 2005. Extraction and classification of facemarks with kernel methods. In *Proc. 10th Int’l Conf. Intelligent User Interfaces*.
- T. Yamada, S. Tsuchiya, S. Kuroiwa, and F. Ren. 2007. Classification of facemarks using n-gram. In *International Conference on Natural Language Processing and Knowledge Engineering*, pages 322–327.