Using the Web to Train a Mobile Device Oriented Japanese Input Method Editor

Xianchao Wu, Rixin Xiao, Xiaoxin Chen
Baidu Inc.
wuxianchao@gmail.com, {xiaorixin, chenxiaoxin}@baidu.com

Abstract

This paper describes the construction of a Japanese Input Method Editor (IME) system for mobile devices, using the large-scale Web pages. We provide the training process of our IME model, n-pos model for local Kana-Kanji conversion and n-gram model for online cloud service. Especially, we propose an online algorithm of mining new compound words, together with the detailed post-filtering process to prune the billion level entries to be used in mobile services. Experiments show that our IME system outperforms two state-of-the-art Japanese IME baselines. We have released our system in a completely free form[1] and the system is currently used by millions of users.

1 Introduction

Mobile devices such as smart phones and tablet PCs are used by billions of users. For example, Google's Android system has obtained more than 900 million[2] active devices till May 2013. In this paper, we use large-scale Web pages to train a Japanese IME system for mobile devices.

Languages such as Chinese and Japanese can not been typed directly using Latin keyboards. This is because there are only 26 English letters in a Latin keyboard, yet the number of Chinese characters are in tens thousands level. Furthermore, by connecting several Chinese characters together, the number of yielded words and frequently used phrases/idioms are in million level. Thus, language-dependent Input Method Editor (IME) systems are indispensable which maps from combinations of English letters to Chinese characters/words/phrases and Japanese Kanji/Kana sequences.

For example, suppose we want to input a Japanese verb "炒める" (means "fry", "炒" is a Japanese Kanji character, "める" are two Japanese Kana characters). We first need to know the Kana pronunciation of the Kanji character "炒", which is "いた". That is, the verb is pronounced as "いためる". Then, we need to know the mapping from English letters to Japanese Kanas. Here, "い", "た", "め", "る" respectively correspond to "i", "ta", "me", "ru", which can be directly typed by using the Latin keyboards. This mapping (e.g., from "i" to "い") is unique and predefined already. Thus, the real challenge for constructing an IME system for Japanese is to provide the most reasonable Kanji sequence from a given Kana sequence:

- one Kanji character can have several correct Kana pronunciations (e.g., "炒" can be pronounced as "いた", "しょう", "そう", etc.);
- one Kana sequence corresponds to numerous Kanji candidates (such as "いためる" for "炒める", "痛める" (pain), etc.);

It is the context that determines the selection of the most reasonable Kanji sequence. For example, "心をいためる" ("心" = heart, "を" is a Japanese particle right follows an argument and before the argument's predicate) requires the Kanji to be "痛める" (heart pain) and "野菜をいためる" needs the Kanji to be "炒める" (fry vegetables). However, it is not a trivial work for modelling the context. That is, how to choice the context from large-scale Web pages such that the context is optimized to be used in a mobile device oriented Japanese IME?

To answer this question, we need to consider the following constraints:

- mobile devices need more strict controlling of CPU and memory usages than laptops;

[1] http://simeji.me/
[2] http://www.android.com/

International Joint Conference on Natural Language Processing, pages 1209–1215,
Nagoya, Japan, 14-18 October 2013.

- free wireless services are not supposed to be available anywhere, any time.

Consequently, we have to limit the number of Kana-Kanji entries to be loaded into memory and ensure a high precision of Kana-Kanji conversion even without on-line services (such as cloud input).

2 The Model

Our Japanese IME system is constructed based on the n-pos[3] model (Mori et al., 1999; Komachi et al., 2008; Kudo et al., 2011). For statistical Kana-Kanji conversion, we predicate the optimal mixed Kana-Kanji sequence $\hat{\mathbf{y}}$ ($= w_1 ... w_n$) from the input Hirakana sequence $\mathbf{x}$:

$$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} P(\mathbf{y}) P(\mathbf{x}|\mathbf{y}) \quad (1)$$

$$P(\mathbf{y}) = \prod_{i=1}^{n} P(w_i|c_i) P(c_i|c_{i-1}) \quad (2)$$

$$P(\mathbf{x}|\mathbf{y}) = \prod_{i=1}^{n} P(r_i|w_i) \quad (3)$$

As shown in Figure 1, for training this model, we used 2.5TB Japanese Web pages as the training data. We run Mecab[4] with IPA dictionary[5] on Hadoop[6], an open source software that implemented the Map-Reduce framework (Dean and Ghemawat, 2004), for parallel word segmenting, Part-of-Speech (POS) tagging, and Kana pronunciation annotating. Then, based on maximum likelihood estimation, we estimate:

- $P(c_i|c_{i-1})$, bi-gram POS tag model;
- $P(w_i|c_i)$, POS-to-word emission model, from c_i to a word w_i; and,
- $P(r_i|w_i)$, pronunciation model, from w_i to its Kana pronunciation r_i.

There are several lexicons/models to be used in the final IME system. The first is called the basic lexicon. An entry in this lexicon is alike $< w_i^{i+m}, c_i^{i+m}, r_i^{i+m} >$. Here, w_i^{i+m} stands for $m+1$ words (of $w_i ... w_{i+m}$). One word w_i exactly corresponds to one POS tag c_i and one Kana

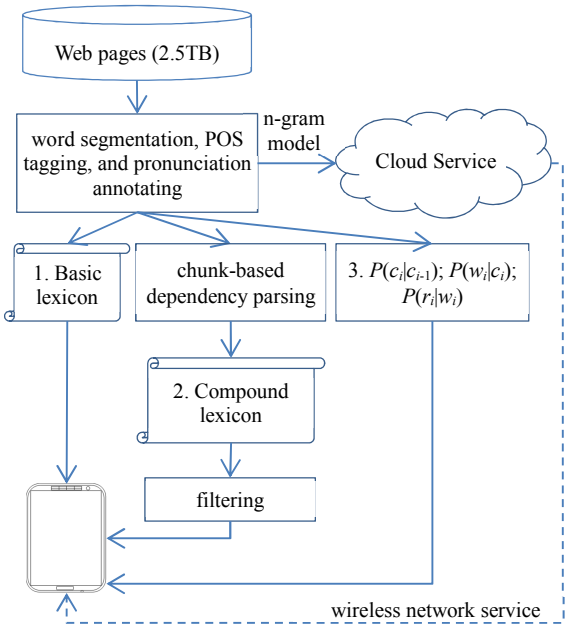

Figure 1: The main process of using the Web to train the IME system.

sequence r_i as its pronunciation. One word sequence with multiple reasonable POS sequences and/or Kana pronunciations will be stored separately as different entries. This lexicon contains: (1) Japanese words (such as particles, adjectives, adverbs, verbs, nouns, etc.) with the highest frequencies, and (2) the most frequently used idioms which are collected manually by our Japanese language experts.

The second is the compound lexicon which contains new words, collocations, and predicate-argument phrases. As drawn in Figure 1, dependency parsing is performed before mining. Web sentences were parsed by a state-of-the-art chunk-based Japanese dependency parser, Cabocha[7] (Kudo and Matsumoto, 2002a). The mining and filtering process will be introduced in the next section. The motivation of constructing this lexicon is to extract the most important context information, such as the strong constraints among predicates and their arguments. For example, as former mentioned, the pre-predicate arguments such as "心" (heart) or "野菜" (vegetables) with given Kana sequence "をいためる" will determine which predicate verb to choose, "痛める" or "炒める".

The third is the n-pos model with three kinds of probabilities which are used during decoding, i.e., searching the n-best $\mathbf{y}$s from a given input Kana

[3] n-pos model is short for n-gram part-of-speech model
[4] https://code.google.com/p/mecab/
[5] http://code.google.com/p/mecab/downloads/detail?name=mecab-ipadic-2.7.0-20070801.tar.gz
[6] http://hadoop.apache.org/
[7] http://code.google.com/p/cabocha/

sequence $\mathbf{x}$.

Finally, we train a 4-gram language model on surface word level and construct a cloud Kana-Kanji conversion service through wireless network communication between a mobile device and the cloud. The only difference with former n-pos model is the factorization of $P(\mathbf{y})$:

$$P(\mathbf{y}) = \prod_{i=1}^{n} P(w_i|w_{i-1}, w_{i-2}, w_{i-3}) \quad (4)$$

The first three lexicons/models are stored in the mobile devices to be accessed during Kana-Kanji decoding using Equation 1. The final 4-gram language model is estimated in a different way from the n-pos model. Thus, we are forced to interpolate cloud's m-best Kanji candidates into local mobile device's n-best Kanji candidates. We perform duplicated candidate removing before interpolating. Possible methods includes:

- insert the cloud candidates into fixed positions, e.g., from the second position to the m+1 position of the local n-best list; or,
- upload the local n-best candidates to the cloud and then use the 4-gram language model to compute the candidates' language model scores; or,
- download POS tags of the m-best candidates from the cloud and locally compute their scores under the local n-pos model.

The first method is the simplest without a large usage of the wireless network. The later two methods make the direction comparison of cloud and local candidates yet possibly take a large usage of the network. For simplicity, we choose the first method (e.g., taking cloud result as the first candidate) in our IME system.

3 Compound Word Mining and Filtering

3.1 Mining Process

The basic lexicon used in our Japanese IME system is short at capturing new words and phrases, which are appearing everyday in the latest Web pages. For example, person names, technical terms and organization names are newly created and used in Web pages such as news, blogs, question-answering systems. We argue it is essential for the IME system to regularly update its compound lexicon to cover these new and hot words/phrases.

Alike the format of the basic lexicon, entries with $m+1$ (m differs among the entries) words in the compound lexicon is also triples of $< w_i^{i+m}, c_i^{i+m}, r_i^{i+m} >$. In this paper, we mine three types of new compound words, together with their pronunciations from Japanese Web pages:

- words, which are combinations of single characters and shorter words (e.g., "副/ふく 垢/あか" = "secondary (twitter) account");
- collocations, which are combinations of words (e.g., "ドバイ ショック" = "Dubai (debt) crisis"). Here, Japanese collocations are allowed to include Kanjis, Katakanas and Hirakanas. Different from many former researches (Manning and Schütze, 1999; Liu et al., 2009) which only mine collocations of two words, we do not limit the number of words in our "collocation" lexicon; and,
- predicate-argument phrases, which are combinations of chunks constrained by semantic dependency relations (e.g., "心 を 痛める" = heart pain).

New words and collocations are mined from single chunks in the dependency trees generated by Cabocha. This mining idea is based on the fact that an Japanese morphological analyser (e.g., Mecab) tends to split one out-of-vocabulary (OOV) word into a sequence of known Kanji characters, and most of these known Kanji characters are annotated to be notional words. Consequently, Cabocha, which takes words/characters and their POS tags as features for discriminative training using a SVM model (Kudo and Matsumoto, 2002b), can still *correctly* tend to include these single-Kanji-character words into one chunk. Thus, we can re-combine the wrongly separated pieces into one (compound) word.

Predicate-argument phrases are mined from adjacent chunks with dependency relations. Since Japanese is a Subject-Object-Verb (SOV) language, the predicate frequently follows its subject/object arguments.

Figure 2 and 3 give the distributions (number of words per compound word vs. the frequency of compound words in a similar number of words) of single/double chunk lexicons (without any filtering yet). For new words and collocations mined

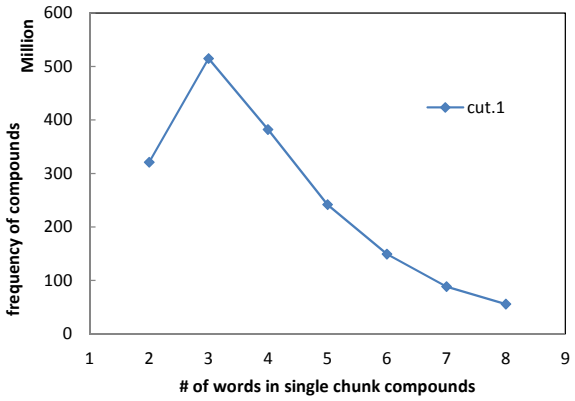

Figure 2: The distributions of the number of words per compound in single chunk lexicon, mined from the 2.5TB web data.

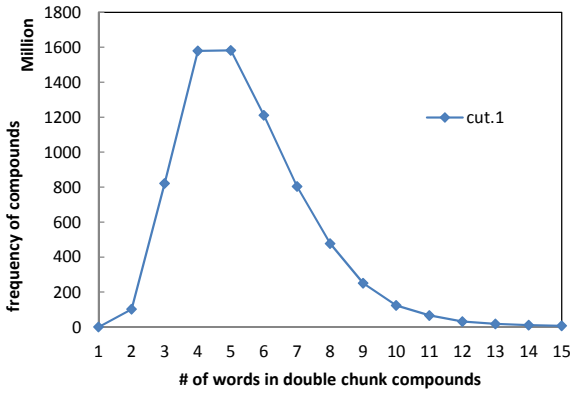

Figure 3: The distributions of the number of words per predicate-argument phrases, mined from the 2.5TB web data.

from single chunks, we limit the number of words frequently ranges from 2 to 8. For predicate-argument phrases, most number (of words) is in the interval of [2, 11].

Frequency information is used for pruning the compound entries to be finally used in mobile devices. The mining algorithm can be performed in an online way. For one aspect, we can timely crawl the latest Web pages and execute the mining process. The frequencies of lastly mined entries can be simply accumulated to the existing entries. On the other hand, we allow the users to upload their input logs to the cloud and execute the mining process to extract single user oriented personally entries. Again, the frequencies of similar words/phrases are simply accumulated.

Recall the n-pos model and the n-gram language model. Since all the probabilities are estimated in a maximum likelihood way, we can simply update the probabilities in these models by accumulating frequencies to similar words/phrases. Thus, we say that our IME system is self-growing as the Web becoming larger and users using it longer.

3.2 Filtering Process

After successful mining of the compound lexicon, it is still challenging to prune it to be used in mobile devices with limited computing ability and memory. The trade-off is that, we have to maintain a good enough local lexicon yet with extremely limited number of entries. We use the following algorithms step by step for filtering:

- use the likelihood ratio method as described in (Manning and Schütze, 1999);
- use the LH score as described in (Okazaki and Ananiadou, 2006);
- use the log file of the cloud service; and,
- use hand-made deep filtering rules.

We hereafter describe these filtering strategies. Likelihood ratio is an approach to hypothesis testing, which has been proved to be appropriate for sparse data (Manning and Schütze, 1999). That is, even for candidate phrases will relatively low frequencies, if they share a strong relation with each other, then they are still possibly kept. Likelihood ratio is simply a number that tells us how much more likely one hypothesis is than the other one:

- Hypothesis 1. $P(w_2|w_1) = p = P(w_2|\neg w_1)$;
- Hypothesis 2. $P(w_2|w_1) = p \neq P(w_2|\neg w_1)$.

The first hypothesis judges if the occurrence of word w_2 is *independence* with word w_1, and the second hypothesis is about *dependence* which is good evidence for an interesting collocation candidate. The computing of $log\lambda = log(L(H_1)/L(H_2))$ exactly follows the definitions in (Manning and Schütze, 1999). Deal to short of space, we skip the detail here. Note that for phrases with more than two words (e.g., three words), we separately take the first/last two words as one unit, and the other word as another unit. Then, if and only if both w_1w_2, w_3 and w_1, w_2w_3 are collocation candidates, then $w_1w_2w_3$ is taken as one reasonable collocation.

After likelihood ratio based filtering, we checked the remaining entities and found too

many nested entries. For example, even both $w_1w_2w_3$ and w_1w_2 were kept in the final compound lexicon, only one of them was judged manually to be the correct one. Dealing with this problem, we use the LH score formula as described in (Okazaki and Ananiadou, 2006). we reuse the heuristic LH formula to compute the collocation likelihood LH(c) for a candidate c:

$$\mathrm{LH}(c) = \mathrm{freq}(c) - \sum_{t \in T_c} \mathrm{freq}(t) \times \frac{\mathrm{freq}(t)}{\sum_{t \in T_c} \mathrm{freq}(t)}.$$

Here, c is a Kanji (sub-)sequence candidate; freq(c) denotes the frequency of co-occurrence of c with the final/first word(s) of the phrases; and T_c is a set of nested Kanji sequence candidates, each of which consists of a preceding (or, succeeding) Kanji or Kana character followed by (or, follows) the candidate c. This LH score can be computed in left-to-right (i.e., taking the former one or more words as no-changing words and seek the word list that follows these former words) direction or right-to-left direction. For example, for left-to-right computing, we can collect all the phrases starts with the similar word w_1 and then collect all the compound entries start with w_1. Then, after computing LH score, we can limit the number of entries start with w_1.

Even after these two automatic filtering algorithms, we still find there are too many entries remaining in the compound lexicon. The third step of filtering is the usage of the cloud log file which stores the entries that users uploads to the cloud. This filtering strategy is to only keep those entries whose Kana pronunciations were found in the log file. The consideration is to connect the Web to the real requirements of the users.

Finally, we manually check the remaining lexicon and construct deep filtering rules. For example, entries that starts with "ない", "等" are pruned out; entries with POS tags of "particles", "auxilary verbs" are pruned out. Note that, this manual checking is performed before the final lexicon is generated. Filtering rules are constructed after this manual checking step and further used for filtering the test set as can be find in the next section (22 entries were filtered from the 5K entries in the test set).

4 Experimental Results

We have described in detail the training and filtering process for constructing lexicons and models.

Systems	top-1	top-6	top-12	Missing Words
Baseline1	84.91	89.11	89.31	532
Baseline2	82.64	94.23	94.80	112
IME-basic	81.36	85.82	85.82	705
+ compound	85.78	91.22	91.30	431
+ cloud (1st)	88.99	94.98	96.44	41

Table 1: The top-1/6/12 precisions (%) of the baselines and our IME system under several configurations. Here, IME-basic stands for our IME system with only the basic lexicon; + compound stands for the system together with the basic lexicon and the compound lexicon; + cloud stands for taking cloud's best candidate as IME's first candidate.

In terms of the decoding algorithm, we use beam searching for n-best Viterbi decoding (Huang and Chiang, 2005). The training data is a 2.5TB Japanese Web page set. Our basic lexicon contains around 100k entries, while the compound lexicon is limited to contain around 50k entries. No limitation is set to the 4-gram language model running in the cloud.

Our test set contains 4,978 Kana-Kanji entries of frequently used word, idioms, and phrases. The entries of this test set comes from the following three lexicons/corpora:

- (partial) "JDMWE" (Japanese Dictionary of Multi-Word Expressions) (Shudo et al., 2011) lexicon with 2,169 entries;
- "Nagoya" compound word lexicon[8] with 3,628 entries such as idioms;
- 16,611 long form words in the "BCCWJ" (Balanced Corpus of Contemporary Written Japanese) corpus (Maekawa, 2008).

We then retrieve each entry in these three lexicons using Google[9] and only keep the top 5K entries with higher frequencies. After obtaining the 5K entries, we perform manually constructed deep filtering rules (which have been used during training) and remove 22 entries which are judged to be not suitable to be taken as collocations with complete meaning.

We use top-n precisions P_n to evaluate the accuracy of the IME systems. We use <k_m, r_m> to

[8] http://kotoba.nuee.nagoya-u.ac.jp/jc2/base/list
[9] https://www.google.co.jp/

express one entry in the test set, where m ranges from 1 to M, k_m is the Kana input and r_m is the Kanji reference.

$$P_n = \frac{\sum_{m=1}^{M}\{\delta(r_m, \text{IME}_n(k_m))\}}{M} \quad (5)$$

Given one k_m, $\text{IME}_n(k_m)$ generates the n-best Kanji candidate for k_m. The $\delta()$ function is defined as follows:

$$\delta(r_m, \text{IME}_n(k_m)) = \begin{cases} 1 & \text{if } r_m \in \text{IME}_n(k_m), \\ 0 & \text{otherwise.} \end{cases}$$

When n takes 1, P_1 is equivalent to the traditional definition of precision.

Table 1 shows the top-n precisions and the number of missing words of two starte-of-the-art Japanese IME baseline systems and our IME system under several configurations of lexicons/models. Both the baseline systems and our IME systems are in mobile device versions.

Here, baseline1[10] is a commercial Japanese IME system whose lexicon contains around 200k entries. This baseline system is constructed by using statistical methods on relatively a small-scale training data and a lot of hand-made Kana-Kanji conversion rules. Deal to resource limitation, we could not obtain further detailed technical information of this system and can only buy one copy and test it in an open testing way.

The second baseline IME system (Kudo et al., 2011), baseline2[11], is constructed in a statistical way by using the large-scale Japanese Web pages as the training data. N-pos model is also the major model supporting its training and decoding algorithms. This system can be freely obtained.

From the table, we have the following observations:

- when only using the basic lexicon, our IME system is worse than both of the baselines;
- when the compound lexicon is appended, the top-1 precision of our IME system is better than baselines, yet top-6/12 precisions are still not good (by checking the lexicon size of baseline2, we found that around 300k to 400k entries were contained. Yet there are only around 100k+50k entries in our basic/compound lexicons);

[10]http://www.justsystems.com/jp/products/atok_android/

[11]https://play.google.com/store/apps/details?id=com.google.android.inputmethod.japanese

Systems	top-1	top-6	top-12	Missing Words
IME	76.12	82.05	82.05	224
+ log	79.41	87.74	87.82	152
improves	3.29	5.69	5.77	-72

Table 2: The top-1/6/12 precisions (%) of our IME system under several configurations. Here, IME stands for the system using basic and compound lexicons; + log stands for appending compound entries mined from users' log.

- finally, by using cloud service, the top-n precisions are significantly better than two baselines.

We did another experiment for testifying the "online" ability of our IME system. The training data is the users' logs. We used these logs (of during two months) to extract compound words and append them to existing compound lexicons. There are 6k entries appended. The testing data (which contains 1,248 entries) is a set of compound words using logs of the latest three days.

Table 2 shows the changes of top-1/6/12 precisions by appending the compound entries mined from users' log. We observe that the precisions are improved 3.29% to 5.77%. These improvements show evidence that the IME system can grow itself in an online way with more data and more users.

5 Conclusion

We have described the construction of a Japanese Input Method Editor (IME) system for mobile devices, using 2.5TB Web pages. We provided the training process of our IME model, n-pos model for local Kana-Kanji conversion and n-gram model for online cloud service. In particular, we described an online algorithm of mining new compound words, together with the detailed post-filtering process to prune the billion level entries to be used in mobile services. Experiments showed that our IME system outperforms two state-of-the-art Japanese IME baselines. We have released our system in a completely free form and the system has been downloaded by more than 5 million users and is currently used by users in million level[12].

[12]https://play.google.com/store/apps/details?id=com.adamrocker.android.input.simeji

References

Jeffrey Dean and Sanjay Ghemawat. 2004. Mapreduce: simplified data processing on large clusters. In *Proceedings of OSDI*.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT*.

Mamoru Komachi, Shinsuke Mori, and Hiroyuki Tokunaga. 2008. Japanese, the ambiguous, and input methods (in japanese). In *Proceedings of the Summer Programming Symposium of Information Processing Society of Japan*.

Taku Kudo and Yuji Matsumoto. 2002a. Japanese dependency analysis using cascaded chunking. In *Proceedings of Co-NLL*, pages 63–69.

Taku Kudo and Yuji Matsumoto. 2002b. Japanese dependency analysis using cascaded chunking. In *Proceedings of CoNLL-2002*, pages 63–69. Taipei, Taiwan.

Taku Kudo, Taiyaki Komatsu, Toshiyuki Hanaoka, Jun Mukai, and Yusuke Tabata. 2011. Mozc: A statistical kana-kanji conversion system (in japanese). In *Proceedings of Japan Natural Language Processing*, pages 948–951.

Zhanyi Liu, Haifeng Wang, Hua Wu, and Sheng Li. 2009. Collocation extraction using monolingual word alignment method. In *Proceedings of EMNLP*, pages 487–495, Singapore, August.

Kikuo Maekawa. 2008. Compilation of the kotonoha-bccwj corpus (in japanese). *Nihongo no kenkyu (Studies in Japanese)*, 4:82–95.

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, Massachusetts, May.

Shinsuke Mori, Masatoshi Tsuchiya, Osamu Yamaji, and Makoto Nagao. 1999. Kana-kanji conversion by a stochastic model (in japanese). *Journal of Information Processing Society of Japan*, 40(7).

Naoaki Okazaki and Sophia Ananiadou. 2006. Building an abbreviation dictionary using a term recognition approach. *Bioinformatics*, 22(22):3089–3095.

Kosho Shudo, Akira Kurahone, and Toshifumi Tanabe. 2011. A comprehensive dictionary of multiword expressions. In *Proceedings of ACL-HLT*, pages 161–170, Portland, Oregon, USA, June.