Mining Transliterations from Web Query Results: An Incremental Approach

Jin-Shea Kuo
Chung-Hwa Telecomm. Laboratories, Taiwan
d8807302 @gmail.com

Haizhou Li
Institute for Infocomm Research, Singapore 119613
hzli@ieee.com

Chih-Lung Lin
Chung Yuan Christian University, Taiwan
linclr@gmail.com

Abstract

We study an adaptive learning framework for phonetic similarity modeling (PSM) that supports the automatic acquisition of transliterations by exploiting minimum prior knowledge about machine transliteration to mine transliterations incrementally from the live Web. We formulate an incremental learning strategy for the framework based on Bayesian theory for PSM adaptation. The idea of incremental learning is to benefit from the continuously developing history to update a static model towards the intended reality. In this way, the learning process refines the PSM incrementally while constructing a transliteration lexicon at the same time on a development corpus. We further demonstrate that the proposed learning framework is reliably effective in mining live transliterations from Web query results.

1 Introduction

Transliteration is a process of rewriting a word from one language into another by preserving its pronunciation in its original language, also known as *translation-by-sound*. It usually takes place between languages with different scripts, for example, from English to Chinese, and words, such as proper nouns, that do not have "easy" or semantic translations.

The increasing size of multilingual content on the Web has made it a live information source rich in transliterations. Research on automatic acquisition of transliteration pairs in batch mode has shown promising results (Kuo *et al*., 2006). In dealing with the dynamic growth of the Web, it is almost impossible to collect and store all its contents in local storage. Therefore, there is a need to develop an incremental learning algorithm to mine transliterations in an on-line manner. In general, an incremental learning technique is designed for adapting a model towards a changing environment. We are interested in deducing the incremental learning method for automatically constructing an English-Chinese (*E-C*) transliteration lexicon from Web query results.

In the deduction, we start with a phonetic similarity model (PSM), which measures the phonetic similarity between words in two different scripts, and study the learning mechanism of PSM in both batch and incremental modes. The contributions of this paper include: (i) the formulation of a batch learning framework and an incremental learning framework for PSM learning; (ii) a comparative study of the batch and incremental unsupervised learning strategies.

In this paper, Section 2 briefly introduces prior work related to machine transliteration. In Section 3, we formulate the PSM and its batch and incremental learning algorithms while in Section 4, we discuss the practical issues in implementation. Section 5 provides a report on the experiments conducted and finally, we conclude in Section 6.

2 Related Work

Much of research on extraction of transliterations has been motivated by information retrieval techniques, where attempts to extracting transliteration pairs from large bodies of corpora have been made. Some have proposed extracting translations from parallel or comparable *bitexts* using co-occurrence analysis or a context-vector approach (Fung and Yee, 1998; Nie *et al.*, 1999). These methods compare the semantic similarities between source and target words without taking their phonetic similarities into account.

Another direction of research is focused on es-

tablishing the phonetic relationship between transliteration pairs. This typically involves the encoding of phoneme- or grapheme-based mapping rules using a generative model trained from a large bilingual lexicon. Suppose that *EW* and *CW* form an *E-C* transliteration pair. The phoneme-based approach (Knight & Graehl, 1998) first converts *EW* into an intermediate phonemic representation and then converts the phonemic representation into its Chinese counterpart *CW*. The grapheme-based approach, also known as *direct orthographical mapping* (Li *et al.*, 2004), which treats transliteration as a statistical machine translation problem under monotonic constraints, has also achieved promising results.

Many efforts have also been channeled to tapping the wealth of the Web for harvesting transliteration/translation pairs. These include studying the query logs (Brill *et al.*, 2001), unrelated corpora (Rapp, 1999), and comparable corpora (Sproat *et al.* 2006). To establish cross-lingual correspondence in the harvest, these algorithms usually rely on one or more statistical clues (Lam *et al.*, 2004), such as the correlation between word frequencies, and cognates of similar spelling or pronunciations. In doing so, two things are needed: first, a robust mechanism that establishes statistical relationships between bilingual words, such as a phonetic similarity model which is motivated by transliteration modeling research; and second, an effective learning framework that is able to adaptively discover new events from the Web.

In Chinese/Japanese/Korean (CJK) Web pages, translated terms are frequently accompanied by their original Latin words, with the Latin words serving as the appositives of the CJK words. In other words, the *E-C* pairs are always closely collocated. Inspired by this observation in CJK texts, some algorithms were proposed (Kuo *et al.*, 2006) to search over the close context of an English word in a Chinese predominant bilingual snippet for transliteration.

Unfortunately, many of the reported works have not taken the dynamic nature of the Web into account. In this paper, we study the learning framework of the phonetic similarity model, which adopts a transliteration modeling approach for transliteration extraction from the Web in an incremental manner.

3 Phonetic Similarity Model

Phonetic similarity model (PSM) is a probabilistic model that encodes the syllable mapping between *E-C* pairs. Let $ES = \{e_1, ...e_m, ...e_M\}$ be a sequence of English syllables derived from *EW* and $CS = \{s_1, ...s_n, ...s_N\}$ be the sequence of Chinese syllables derived from *CW*, represented by a Chinese character string $CW \rightarrow w_1, ...w_n, ..., w_N$. If each of the English syllables is drawn from a vocabulary of *X* entries, $e_m \in \{x_1, ..., x_I\}$, and each of the Chinese syllable from a vocabulary of *Y* entries, $s_n \in \{y_1, ..., y_J\}$, then the *E-C* transliteration can be considered as a generative process formulated by the noisy channel model, which recovers the input *CW* from the observed output *EW*. Applying Bayesian rule, we have Eq. (1), where $P(EW \mid CW)$ is estimated to characterize the noisy channel, known as the transliteration probability and $P(CW)$ is a language model to characterize the source language.

$$P(CW \mid EW) = P(EW \mid CW)P(CW) / P(EW). \quad (1)$$

Following the *translation-by-sound* principle, $P(EW \mid CW)$ can be approximated by the phonetic probability $P(ES \mid CS)$, which is given by Eq. (2).

$$P(ES \mid CS) = \max_{\Delta \in \Gamma} P(ES, \Delta \mid CS), \quad (2)$$

where Γ is the set of all possible alignment paths between *ES* and *CS*. To find the best alignment path Δ, one can resort to a dynamic warping algorithm (Myers and Rabiner, 1981). Assuming conditional independence of syllables in *ES* and *CS*, we have $P(ES \mid CS) = \prod_{k=1}^{K} P(e_{m_k} \mid s_{n_k})$ where *k* is the index of alignment. We rewrite Eq.(1) as,

$$P(CW \mid EW) \approx P(ES \mid CS)P(CW) / P(EW). \quad (3)$$

The language model $P(CW)$ in Eq.(3) can be represented by the *n*-gram statistics of the Chinese characters derived from a monolingual corpus. Using bigram to approximate the n-gram model, we have

$$P(CW) \approx P(w_1) \prod_{n=2}^{N} P(w_n \mid w_{n-1}). \quad (4)$$

Removing $P(EW)$ from Eq.(3) which is not a function of *CW*, a PSM Θ now consists of both $P(ES \mid CS)$ and $P(CW)$ parameters (Kuo *et al.*, 2007). We now look into the mathematic formulation for the learning of $P(ES \mid CS)$ parameters from a bilingual transliteration lexicon.

3.1 Batch Learning of PSM

A collection of manually selected or automatically extracted *E-C* pairs can form a transliteration lexicon. Given such a lexicon for training, the PSM parameters can be estimated in a batch mode. An initial PSM is bootstrapped using limited prior knowledge such as a small amount of transliterations, which may be obtained by exploiting co-occurrence information (Sproat *et al*., 2006). Then we align the *E-C* pairs using the PSM Θ and derive syllable mapping statistics.

Suppose that we have the event counts $c_{i,j} = count(e_m = x_i, s_n = y_j)$, and $c_j = count(s_n = y_j)$ for a given transliteration lexicon D with alignments Λ. We would like to find the parameters $P_{i|j} = P(e_m = x_i \mid s_n = y_j)$, $< e_m, s_n > \in \Lambda$, that maximize the probability,

$$P(D, \Lambda \mid \Theta) = \prod_{\Lambda} P(e_m \mid s_n) = \prod_j \prod_i P_{i|j}^{c_{i,j}}, \quad (5)$$

where $\Theta = \{P_{i|j}, i = 1,...,I, j = 1,...,J\}$, with maximum likelihood estimation (MLE) criteria, subject to the constraints of $\sum_i P_{i|j} = 1, \forall j$. Rewriting Eq.(5) in log-likelihood (LL)

$$LL(D, \Lambda \mid \Theta) = \sum_{\Lambda} \log P(e_m \mid s_n) = \sum_j \sum_i c_{i,j} \log P_{i|j} \quad (6)$$

It is described as the *cross-entropy* of the true data distribution $c_{i,j}$ with regard to the PSM model. Given an alignment $\Delta \in \Lambda$, the MLE estimate of PSM is:

$$P_{i|j} = c_{i,j} / c_j . \quad (7)$$

With a new PSM, one is able to arrive at a new alignment. This is formulated as an *expectation-maximization* (EM) process (Dempster, 1977), which assumes that there exists a mapping $D \to \Lambda$, where Λ is introduced as the latent information, also known as *missing* data in the EM literature. The EM algorithm maximizes the likelihood probability $P(D \mid \Theta)$ over Θ by exploiting $P(D \mid \Theta) = \sum_{\Lambda} P(D, \Lambda \mid \Theta)$.

The EM process guarantees non-decreasing likelihood probability $P(D \mid \Theta)$ through multiple EM steps until it converges. In the E-step, we derive the event counts $c_{i,j}$ and c_j by force-aligning all the *E-C* pairs in the training lexicon D using a PSM. In the M-step, we estimate the PSM parameters Θ by Eq.(7). The EM process also serves as a refining process to obtain the best alignment between the *E-C* syllables. In each EM cycle, the model is updated after observing the whole corpus D. An EM cycle is also called an iteration in batch learning. The batch learning process is described as follows and depicted in Figure 1.

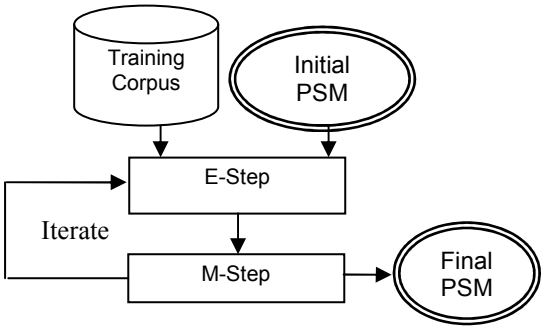

Figure 1. Batch learning of PSM

***Batch Learning Algorithm*:**

Start: Bootstrap PSM parameters $P_{i|j}$ using prior phonetic mapping knowledge;

E-Step: Force-align corpus D using $P_{i|j}$ to obtain Λ and hence the counts of $c_{i,j}$ and c_j;

M-Step: Re-estimate $P_{i|j} = c_{i,j} / c_j$ using the counts from E-Step;

Iterate: Repeat E-Step and M-Step until $P(D \mid \Theta)$ converges;

3.2 Incremental Learning of PSM

In batch learning all the training samples have to be collected in advance. In a dynamically changing environment, such as the Web, new samples always appear and it is impossible to collect all of them. Incremental learning (Zavaliagkos, 1995) is devised to achieve rapid adaptation towards the working environment by updating the model as learning samples arrive in sequence. It is believed that if the statistics for the E-step are incrementally collected and the parameters are frequently estimated, incremental learning converges quicker because the information from the new data contributes to the parameter estimation more effectively than the batch algorithm does (Gotoh *et al*., 1998). In incremental learning, the model is typically updated progressively as the training samples become available and the number of incremental samples may vary from as few as one to as many as they are available. In the extreme case where all the learning samples are

available and the updating is done after observing all of them, the incremental learning becomes batch learning. Therefore, the batch learning can be considered as a special case of the incremental learning.

The incremental learning can be formulated through maximum *a posteriori* (MAP) framework, also known as Bayesian learning, where we assume that the parameters Θ are random variables subject to a prior distribution. A possible candidate for the prior distribution of $P_{i|j}$ is the Dirichlet density over each of the parameters $P_{i|j}$ (Bacchiani *et al.*, 2006). Let $\Theta_j = \{P_{i|j}, i = 1,...,I\}$, we introduce,

$$P(\Theta_j) \propto \prod_i P_{i|j}^{\alpha h_{i|j}-1}, \quad \forall j, \tag{8}$$

where $\sum_i h_{i|j} = 1$, and α, which can be empirically set, is a positive scalar. Assuming H is the set of hyperparameters, we have as many hyperparameters $h_{i|j} \in H$ as the parameters $P_{i|j}$. The probability of generating the aligned transliteration lexicon is obtained by integrating over the parameter space, $P(D) = \int P(D|\Theta)P(\Theta)d\Theta$.

This integration can be easily written down in a closed form due to the conjugacy between Dirichlet distribution $\prod_i P_{i|j}^{\alpha h_{i|j}-1}$ and the multinomial distribution $\prod_i P_{i|j}^{c_{i,j}}$. Instead of finding Θ that maximizes $P(D|\Theta)$ with MLE, we maximize *a posteriori* (MAP) probability as follows:

$$\begin{aligned}\Theta_{MAP} &= \arg\max_{\Theta} P(\Theta|D) = \arg\max_{\Theta} P(D|\Theta)P(\Theta)/P(D) \\ &= \arg\max_{\Theta} P(D|\Theta)P(\Theta)\end{aligned} \tag{9}$$

The MAP solution uses a distribution to model the uncertainty of the parameters Θ, while the MLE gives a point estimation (Jelinek, 1990; MacKay, 1994). We rewrite Eq.(9) as Eq.(10) using Eq.(5) and Eq.(8).

$$\Theta_j^{map} \approx \arg\max_{\Theta_j} \prod_i P_{i|j}^{c_{i,j}+\alpha h_{i|j}-1} \tag{10}$$

Eq.(10) can be seen as a Dirichlet function of Θ given H, or a multinomial function of H given Θ. With given prior H, the MAP estimation is therefore similar to the MLE problem which is to find the mode of the kernel density in Eq.(10).

$$P_{i|j} = \lambda h_{i|j} + (1-\lambda) f_{i|j}, \tag{11}$$

where $f_{i|j} = c_{i,j}/c_j$, $\lambda = \alpha/(\sum_i c_{i,j} + \alpha)$.

One can find that λ serves as a weighting factor between the prior and the current observations. The difference between MLE and MAP strategy lies in the fact that MAP introduces prior knowledge into the parameter updating formula. Eq.(11) assumes that the prior parameters H are known and static while the training samples are available all at once.

The idea of incremental learning is to benefit from the continuously developing history to update the static model towards the intended reality. As is often the case, the Web query results in an *on-line* application arrive in sequence. It is of practical use to devise such an incremental mechanism that adapts both parameters and the prior knowledge over time. The *quasi-Bayesian* (QB) learning method offers a solution to it (Bai and Li, 2006).

Let's break up a training corpus D into a sequence of sample subsets $D = \{D_1, D_2, ..., D_T\}$ and denote an accumulated sample subset $D^{(t)} = \{D_1, D_2, ..., D_t\}, 1 \le t \le T$ as an incremental corpus. Therefore, we have $D = D^{(T)}$. The QB method approximates the posterior probability $P(\Theta|D^{(t-1)})$ by the closest tractable prior density $P(\Theta|H^{(t-1)})$ with $H^{(t-1)}$ evolved from historical corpus $D^{(t-1)}$,

$$\begin{aligned}\Theta_{QB}^{(t)} &= \arg\max_{\Theta} P(\Theta|D^{(t)}) \\ &\approx \arg\max_{\Theta} P(D_t|\Theta)P(\Theta|D^{(t-1)}) \\ &= \arg\max_{\Theta} \prod_{i=1}^{I} P_{i|j}^{c_{i,j}+\alpha h_i^{(t-1)}-1}, \quad \forall j.\end{aligned} \tag{12}$$

QB estimation offers a recursive learning mechanism. Starting with a hyperparameter set $H^{(0)}$ and a corpus subset D_1, we estimate $H^{(1)}$ and $\Theta_{QB}^{(1)}$, then $H^{(2)}$ and $\Theta_{QB}^{(2)}$ and so on until $H^{(t)}$ and $\Theta_{QB}^{(t)}$ as observed samples arrive in sequence. The updating of parameters can be iterated between the reproducible prior and posterior estimates as in Eq. (13) and Eq. (14). Assuming $T \to \infty$, we have the following:

Incremental Learning Algorithm:

Start: Bootstrap $\Theta_{QB}^{(0)}$ and $H^{(0)}$ using prior phonetic mapping knowledge and set $t = 1$;

E-Step: Force-align corpus subset D_t using $\Theta_{QB}^{(t-1)}$, compute the event counts $c_{i,j}^{(t)}$ and reproduce prior parameters $H^{(t-1)} \to H^{(t)}$.

$$h_{i|j}^{(t)} = h_{i|j}^{(t-1)} + c_{i,j}^{(t)}/\alpha \tag{13}$$

M-Step: Re-estimate parameters $H^{(t)} \to \Theta_{QB}^{(t)}$ and $P_{i|j}$ using the counts from E-Step.

$$P_{i|j}^{(t)} = h_{i|j}^{(t)} / \sum_i h_{i|j}^{(t)} \quad (14)$$

EM cycle: Repeat E-Step and M-Step until $P(\Theta | D^{(t)})$ converges.

Iterate: Repeat T EM cycles covering the entire data set D in an iteration.

The algorithm updates the PSM as training samples become available. The scalar factor α can be seen as a forgetting factor. When α is big, the update of hyperparameters favors the prior. Otherwise, current observation is given more attention. As for the sample subset size $|D_t|$, if we set $|D_t| = 100$, each EM cycle updates Θ after observing every 100 samples. To be comparable with batch learning, we define an iteration here to be a sequence of EM cycles that covers the whole corpus D. If corpus D has a fixed size $|D^{(T)}|$, an iteration means T EM cycles in incremental learning.

4 Mining Transliterations from the Web

Since the Web is dynamically changing and new transliterations come out all the time, it is better to mine transliterations from the Web in an incremental way. Words transliterated by closely observing common guidelines are referred to as *regular* transliterations. However, in Web publishing, translators in different regions may not observe the same guidelines. Sometimes they skew the transliterations in different ways to introduce semantic implications, also known as wordplay, resulting in *casual* transliterations. *Casual* transliteration leads to multiple Chinese transliteration variants for the same English word. For example, "Disney" may be transliterated into "迪士尼/Di-Shi-Ni/[1]", "迪斯耐/Di-Si-Nai/" and "狄斯耐/Di-Si-Nai/".

Suppose that a sufficiently large, manually validated transliteration lexicon is available, a PSM can be built in a supervised manner. However, this method hinges on the availability of such a lexicon. Even if a lexicon is available, the derived model can only be as good as what the training lexicon offers. New transliterations, such as *casual* ones, may not be well handled. It is desirable to adapt the PSM as new transliterations become available, also referred to as the *learning-at-work* mechanism. Some solutions have been proposed recently along this direction (Kuo *et al.*, 2006). However, the effort was mainly devoted to mitigating the need of manual labeling. A dynamic *learning-at-work* mechanism for mining transliterations has not been well studied.

Here we are interested in an unsupervised learning process, in which we adapt the PSM as we extract transliterations. The *learning-at-work* framework is illustrated in Figure 2. As opposed to a manually labeled training corpus in Figure 1, we insert into the EM process an automatic transliteration extraction mechanism, *search and rank*, as shown in the left panel of Figure 2. The *search and rank* shortlists a set of transliterations from the Web query results or bilingual snippets.

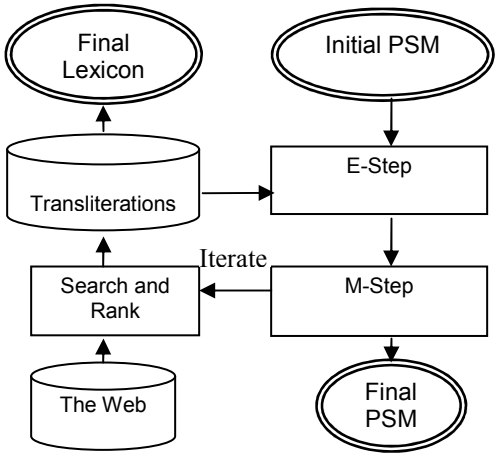

Figure 2. Diagram of unsupervised transliteration extraction – *learning-at-work*.

4.1 Search and Rank

We obtain bilingual snippets from the Web by iteratively submitting queries to the Web search engines (Brin and Page, 1998). Qualified sentences are extracted from the results of each query. Each qualified sentence has at least one English word.

Given a qualified sentence, first, the competing Chinese transliteration candidates are denoted as a set Ω, from which we would like to pick the most likely one. Second, we would like to know if there is indeed a Chinese transliteration *CW* in the close context of the English word *EW*.

We propose ranking the candidates using the PSM model to find the most likely *CW* for a given *EW*. The *CW* candidate that gives the highest posterior probability is considered the most probable

[1] The Chinese words are romanized in Hanyu Pinyin.

candidate CW'.

$$CW' = \arg\max_{CW \in \Omega} P(CW \mid EW) \approx \arg\max_{CW \in \Omega} P(ES \mid CS) P(CW) \quad (15)$$

The next step is to examine if CW' and EW indeed form a genuine *E-C* pair. We define the confidence of the *E-C* pair as the posterior odds similar to that in a hypothesis test under the Bayesian interpretation. We have H_0, which hypothesizes that CW' and EW form an *E-C* pair, and H_1, which hypothesizes otherwise, and use posterior odd σ (Kuo *et al.*, 2006) for hypothesis tests.

Our *search and rank* formulation can be seen as an extension to a prior work (Brill *et al.*, 2001). The posterior odd σ is used as the confidence score so that *E-C* pairs extracted from different contexts can be directly compared. In practice, we set a threshold for σ to decide on a cutoff point for *E-C* pairs short-listing. In this way, the *search and rank* is able to retrieve a collection of transliterations from the Web given a PSM.

4.2 Unsupervised Learning Strategy

Now we can carry out PSM learning as formulated in Section 3 using the transliterations as if they were manually validated. By unsupervised batch learning, we mean to re-estimate the PSM after *search and rank* over the whole database, i.e., in each iteration. Just as in supervised learning, one can expect the PSM performance to improve over multiple iterations. We report the *F*-measure at each iteration. The extracted transliterations also form a new training corpus in next iteration.

In contrast to the batch learning, incremental learning updates the PSM parameters as the training samples arrive in sequence. This is especially useful in Web mining. With the QB incremental optimization, one can think of an EM process that continuously re-estimates PSM parameters as the Web crawler discovers new "territories". In this way, the *search and rank* process gathers qualified training samples D_t after crawling a portion of the Web. Note that the incremental EM process updates parameters more often than batch learning does. To evaluate performance of both learning, we define an iteration to be T EM cycles in incremental learning on a training corpus $D = D^{(T)}$ as discussed in Section 3.2.

5 Experiments

To obtain the ground truth for performance evaluation, each possible transliteration pair is manually checked based on the following criteria: (i) only the phonetic transliteration is extracted to form a transliteration pair; (ii) multiple *E-C* pairs may appear in one sentence; (iii) an *EW* can have multiple valid Chinese transliterations and vice versa. The validation process results in a collection of qualified *E-C* pairs, also referred to as *distinct qualified transliteration pairs* (DQTPs), which form a transliteration lexicon.

To simulate the dynamic Web, we collected a Web corpus, which consists of about 500 MB of Web pages, referred to as SET1. From SET1, 80,094 qualified sentences were automatically extracted and 8,898 DQTPs were further selected with manual validation.

To establish a reference for performance benchmarking, we first initialize a PSM, referred to as *seed* PSM hereafter, using randomly selected 100 seed DQTPs. By exploiting the *seed* PSM on all 8,898 DQTPs, we train a PSM in a supervised batch mode and improve the PSM on SET1 after each iteration. The performance defined below in precision, recall and *F*-measure in the 6th iteration is reported in Table 1 and the *F*-measure is also shown in Figure 3.

$$precision = \#extracted\_DQTPs / \#extracted\_pairs,$$
$$recall = \#extracted\_DQTPs / \#total\_DQTPs,$$
$$F-measure = 2 \times recall \times precision / (recall + precision)$$

	Precision	Recall	F-measure
Closed-test	0.834	0.663	0.739

Table 1. The performance achieved by supervised batch learning on SET1.

We use this closed test (supervised batch learning) as the reference point for unsupervised experiments. Next we further implement two PSM learning strategies, namely unsupervised batch and unsupervised incremental learning.

5.1 Unsupervised Batch Learning

We begin with the same *seed* PSM. However, we use transliterations that are extracted automatically instead of manually validated DQTPs for training. Note that the transliterations are extracted and collected at the end of each iteration. It may differ from one iteration to another. After re-estimating

the PSM in each iteration, we evaluate performance on SET1.

Comparing the two batch mode learning strategies in Figure 3, it is observed that learning substantially improves the *seed* PSM after the first iteration. Without surprise, the supervised learning consistently outperforms the unsupervised one, which reaches a plateau at 0.679 F-measure. This performance is considered as the baseline for comparison in this paper. The unsupervised batch learning presented here is similar to that in (Kuo *et al.*, 2006).

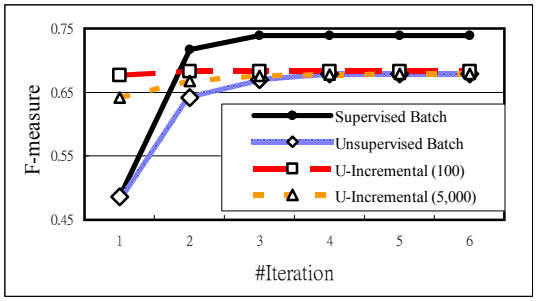

Figure 3. Comparison of F-measure over iterations (U-Incremental: Unsupervised Incremental).

5.2 Unsupervised Incremental Learning

We now formulate an *on-line*[2] unsupervised incremental learning algorithm:

(i) Start with the *seed* PSM, set $t = 1$;

(ii) Extract $|D_t|$ quasi-transliterations pairs followed by E-Step in incremental learning algorithm;

(iii) Re-estimate PSM using $|D_t|$ (M-Step), $t = t + 1$;

(iv) Repeat (ii) and (iii) to crawl over a corpus.

To simulate the *on-line* incremental learning just described, we train and test on SET1 because of the availability of gold standard and comparison with performance by batch mode. We empirically set $\alpha = 0.5$ and study different $|D_t|$ settings. An iteration is defined as multiple cycles of steps (ii)-(iii) that screen through the whole SET1 once. We run multiple iterations.

The performance of incremental learning with $|D_t| = 100$ and $|D_t| = 5,000$ are reported in Figure 3. It is observed that incremental learning benefits from more frequent PSM updating. With $|D_t| = 100$, it not only attains good *F*-measure in the first iteration, but also outperforms that of unsupervised batch learning along the EM process. The PSM updating becomes less frequent for larger $|D_t|$. When $|D_t|$ is set to be the whole corpus, then incremental learning becomes a batch mode learning, which is evidenced by $|D_t| = 5,000$ and it performs close to the batch mode learning. The experiments in Figure 3 are considered closed tests. Next we move on to an actual *on-line* experiment.

5.3 Learning from the Live Web

In practice, it is possible to extract bilingual snippets of interest by repeatedly submitting queries to the Web. With the *learning-at-work* mechanism, we can mine the query results for up-to-date transliterations in an *on-line* environment. For example, by submitting "Amy" to search engines, we may get "Amy-愛咪/Ai-Mi/" and, as a by-product, "Jessica-潔西卡/Jie-Xi-Ka/" as well. In this way, new queries can be generated iteratively, thus new pairs are discovered.

Following the unsupervised incremental learning algorithm, we start the crawling with the same *seed* PSM as in Section 5.2. We adapt the PSM as every 100 quasi-transliterations are extracted, i.e. $|D_t| = 100$. The crawling stops after accumulating 67,944 Web pages, where there are 100 snippets at most in a page, with 2,122,026 qualified sentences. We obtain 123,215 distinct *E-C* pairs when the crawling stops. For comparison, we also carry out unsupervised batch learning over the same 2,122,026 qualified sentences in a single iteration under such an *on-line* environment. As the gold standard for this live corpus is not available, we randomly select 500 quasi-transliteration pairs for manual checking of precision (see Table 2). It is found that incremental learning is more productive than batch learning in discovering transliteration pairs. This finding is consistent with the test results on SET1.

	Unsupervised Batch	Unsupervised Incremental
#distinct *E-C* pairs	67,708	123,215
Estimated Precision	0.758	0.768

Table 2. Comparison between the unsupervised batch and incremental learning from live Web.

The live Web corpus was used in transliteration extraction using active learning (Kuo *et al.*, 2006).

[2] In an actual *on-line* environment, we are not supposed to store documents, thus no iteration can take place.

Kuo *et al*. reported slightly better performance by annotating some samples manually and adapting the learning process in a batch manner. However, it is apparent that, in an *on-line* environment, the unsupervised learning is more suitable for discovering knowledge without resorting to human annotation; incremental learning is desirable as it does not require storing all documents in advance.

6 Conclusions

We have proposed a learning framework for mining *E-C* transliterations using bilingual snippets from a live Web corpus. In this *learning-at-work* framework, we formulate the PSM learning method and study strategies for PSM learning in both batch and incremental manners. The batch mode learning benefits from multiple iterations for improving performance, while the unsupervised incremental one, which does not require all the training data to be available in advance, adapts to the dynamically changing environment easily without compromising the performance. Unsupervised incremental learning provides a practical and effective solution to transliteration extraction from query results, which can be easily extended to other Web mining applications.

References

S. Bai and H. Li. 2006. Bayesian Learning of N-gram Statistical Language Modeling, In *Proc. of ICASSP*, pp. 1045-1048.

M. Bacchiani, B. Roark, M. Riley and R. Sproat. 2006. MAP Adaptation of Stochastic Grammars, Computer Speech and Language, 20(1), pp. 41-68.

E. Brill, G. Kacmarcik, C. Brockett. 2001. Automatically Harvesting Katakana-English Term Pairs from Search Engine Query Logs, In *Proc. of NLPPRS*, pp. 393-399.

S. Brin and L. Page. 1998. The Anatomy of a Large-scale Hypertextual Web Search Engine, In *Proc. of 7th WWW,* pp. 107-117.

A. P. Dempster, N. M. Laird and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm, Journal of the Royal Statistical Society, Ser. B. Vol. 39, pp. 1-38.

P. Fung and L.-Y. Yee. 1998. An IR Approach for Translating New Words from Nonparallel, Comparable Texts, In *Proc. of 17th COLING and 36th ACL*, pp. 414-420.

Y. Gotoh, M. M. Hochberg and H. F. Silverman. 1998. Efficient Training Algorithms for HMM's Using Incremental Estimation, IEEE Trans. on Speech and Audio Processing, 6(6), pp. 539-547.

F. Jelinek. 1999. Self-organized Language Modeling for Speech Recognition, Readings in speech recognition, Morgan Kaufmann, pp. 450-506.

D. Jurafsky and J. H. Martin. 2000. Speech and Language Processing, pp. 102-120, Prentice-Hall, New Jersey.

K. Knight and J. Graehl. 1998. Machine Transliteration, Computational Linguistics, 24(4), pp. 599-612.

J.-S. Kuo, H. Li and Y.-K. Yang. 2006. Learning Transliteration Lexicons from the Web, In *Proc. of 44th ACL*, pp. 1129-1136.

J.-S. Kuo, H. Li and Y.-K. Yang. 2007. A Phonetic Similarity Model for Automatic Extraction of Transliteration Pairs, ACM TALIP, 6(2), pp. 1-24.

H. Li, M. Zhang and J. Su. 2004. A Joint Source Channel Model for Machine Transliteration, In *Proc. of 42nd ACL*, pp. 159-166.

W. Lam, R.-Z. Huang and P.-S. Cheung. 2004. Learning Phonetic Similarity for Matching Named Entity Translations and Mining New Translations, In *Proc. of 27th ACM SIGIR*, pp. 289-296.

D. MacKay and L. Peto. 1994. A Hierarchical Dirichlet Language Model, Natural Language Engineering, 1(3), pp.1-19.

C. S. Myers and L. R. Rabiner. 1981. A Comparative Study of Several Dynamic Time-warping Algorithms for Connected word Recognition, The Bell System Technical Journal, 60(7), pp. 1389-1409.

J.-Y. Nie, P. Isabelle, M. Simard and R. Durand. 1999. Cross-language Information Retrieval based on Parallel Texts and Automatic Mining of Parallel Text from the Web, In *Proc. of 22nd ACM SIGIR*, pp. 74-81.

R. Rapp. 1999. Automatic Identification of Word Translations from Unrelated English and German Corpora, In *Proc. of 37th ACL*, pp. 519-526.

R. Sproat, T. Tao and C. Zhai. 2006. Named Entity Transliteration with Comparable Corpora, In *Proc. of 44th ACL*, pp. 73-80.

G. Zavaliagkos, R. Schwartz, and J. Makhoul. 1995. Batch, Incremental and Instantaneous Adaptation Techniques for Speech Recognition, In *Proc. of ICASSP*, pp. 676-679.