Generic Soft Pattern Models for Definitional Question Answering

Hang Cui Min-Yen Kan Tat-Seng Chua
Department of Computer Science
School of Computing
National University of Singapore
{cuihang, kanmy, chuats}@comp.nus.edu.sg

ABSTRACT

This paper explores probabilistic lexico-syntactic pattern matching, also known as *soft pattern matching*. While previous methods in soft pattern matching are ad hoc in computing the degree of match, we propose two formal matching models: one based on bigrams and the other on the Profile Hidden Markov Model (PHMM). Both models provide a theoretically sound method to model pattern matching as a probabilistic process that generates token sequences. We demonstrate the effectiveness of these models on definition sentence retrieval for definitional question answering. We show that both models significantly outperform state-of-the-art manually constructed patterns. A critical difference between the two models is that the PHMM technique handles language variations more effectively but requires more training data to converge. We believe that both models can be extended to other areas where lexico-syntactic pattern matching can be applied.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Retrieval Models;

General Terms

Algorithms, Measurement, Experimentation

Keywords

Definitional Question Answering, Soft Pattern, Probabilistic Models

1. INTRODUCTION

Natural language texts often exhibit patterns, and thus lexico-syntactic patterns are pervasive in natural language retrieval and extraction tasks, such as information extraction (*e.g.,* [11]). An example of such a pattern is "*<PersonIN> , NNP, BE$ named to POST$*"[1], which may be used to extract the person name *Bob Lloyd* from the sentence "*…<PersonIN> Bob Lloyd </PersonIN>, president and chief operating officer, was named to the <POST> chief executive </POST>*". Besides information extraction, such patterns have been applied to areas including:

1. Question answering (QA): Pattern matching is utilized to improve precision in both factoid QA [12] and definitional QA [19, 6, 1]. The former learns surface text patterns to extract exact answers for simple questions about facts while the latter utilizes more complicated definition patterns to identify definition sentences to define a topic.
2. Retrieval of subjective expressions: Riloff and Wiebe [13] applied an IE system to learn patterns of subjective expressions so that opinions can be identified from news articles.

Lexico-syntactic patterns, which are either manually constructed or machine learned, are often represented and matched as regular expressions. They perform slot by slot matching against test sentences, which we call *hard matching*. While these patterns are highly precise, they often fare poorly in recall because of language variations. For instance, the sample pattern given earlier cannot match the sentence:

<PersonIN> Lee Abraham </PersonIN>, 65 years old, former chairman and chief executive officer of Associated Merchandising Corp., New York, was named to the board of the footwear manufacturer.

which can be reduced to:

<PersonIN> , NUM$ NN ADJ, NNP, NNP, BE$ named to <POST>

The pattern fails to match the sentence due to additional tokens (underscored in the above) that are not found in training samples. Such mismatches are common in natural language texts because authors can use diverse expressions to convey the same meaning. We conjecture that current pattern matching applications may be hindered due to the rigidity of hard matching. One promising technique to circumvent this is *soft pattern matching*. Previously examined by Cui *et al.* [2], soft patterns (SP) have shown to significantly outperform hard patterns in extracting definition sentences as they model language variations probabilistically. While that work has demonstrated the performance of soft patterns, it has not been anchored in a theoretically sound framework.

In this paper, we build upon the earlier work in soft pattern matching. Different from previous empirical work, we show how soft pattern matching is achieved within the framework of two standard probabilistic models. We take both patterns and test instances as sequences of lexical and syntactic tokens. Here, pattern matching can be considered probabilistic generation of

SIGIR'05, August 15–19, 2005, Salvador, Brazil.

Copyright 2005 ACM 1-59593-034-5/05/0008...$5.00.

1 NNP is a POS tag that represents a noun phrase. BE$ stands for all "be" verbs. POST$ refers to the position involved in the management succession.

test sequences based on training sequences. The first model is derived from a bigram language model with linear interpolation [7] of unigram and bigram probabilities. The second model is based on the Profile Hidden Markov Model (PHMM). Parameters in both models are estimated using Expectation Maximization (EM). While the language model and the PHMM have been studied in other areas, their use in modeling lexico-syntactic pattern matching is a novel contribution of our work.

We choose the task of definition sentence retrieval within a typical definitional QA system to demonstrate the models' effectiveness. The reason is two-fold: (1) Definition sentences are diversified in exhibited patterns [2], and thus require flexible pattern matching to achieve high recall; and (2) definitional QA remains one of the least explored areas in QA research. To answer definition questions, such as "*Who is Aaron Copland*" or "*What are prions"*, a system is expected to generate a summary of all pieces of *salient information* (or nuggets) about the given target. To generate definitions for a target, a typical definitional QA system first retrieves linguistic constructs (*e.g.,* sentences, appositives or relative clauses) that might contain salient information about the target, and then summarizes these units into readable definitions. To identify such constructs, current systems use definition patterns to recognize definitions. Although a number of other techniques influence the retrieval of such definition units, component evaluations [2, 19] suggest that definition patterns are the most important feature.

In addition to our theoretical work, we also assess the performance of the formal soft matching models by empirical evaluation. We conduct a series of extrinsic experiments using the two soft pattern models on TREC definitional QA task test data. Our experiments show that the performance of both models significantly outperform state-of-the-art manually constructed hard matching patterns by 11.67% and 9.18% in automatic ROUGE score, and by 9.83% and 7.30% in the manual F_3 measure used by TREC, respectively. Moreover, the two models achieve better performance compared to the original soft matching method proposed in Cui *et al.* [2]. The evaluation results also reveal that the PHMM model is more tolerant to variations in model length but requires more training data for it to converge.

In Section 2, we present a typical definitional QA system with which our experiments are performed, and we give the necessary background for understanding the proposed models. In Section 3, we discuss the two proposed generic soft matching models and our adaptation in detail. We present the evaluation results in Section 4 and conclude the paper with future work in Section 5.

2. SYSTEM ARCHITECTURE AND BACKGROUND

The recent development of definitional QA has been boosted by the Text REtrieval Conference (TREC). TREC-12 and TREC-13 conducted systematic evaluations for definition questions. We use a standard definitional QA system that structurally conforms to those top performing TREC systems [19, 20]. Its architecture is illustrated in Figure 1.

Document Retrieval: Given a definition question such as "Who is X" or "What is X", the search target, i.e., "X" in the question, is fed as query into a standard document retrieval system. The retrieved documents are split into sentences.

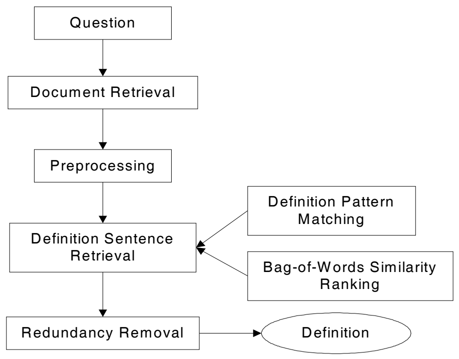
Question
Document Retrieval
Preprocessing
Definition Sentence Retrieval
Definition Pattern Matching
Bag-of-Words Similarity Ranking
Redundancy Removal
Definition

Figure 1. Architecture of the definitional QA system.

Preprocessing of Sentences: We then process the sentences into pattern instances, on which soft pattern generation and matching are performed. First, words specific to the search targets are replaced with their general syntactic (POS or chunk) tags. Remaining words are stemmed. We refer to these remaining lexical words and substituted general syntactic tags as tokens. Definition patterns are expressed by sequences of such tokens. Second, we crop the windows surrounding the search target as instances according to model length *L.* The following box illustrates the process:

The channel Iqra is owned by the Arab Radio and Television company and is the brainchild of the Saudi millionaire, Saleh Kamel.

↓ *POS tagging, chunking and substitution*

DT$ NN <SCH_TERM> BE$ owned by DT$ NP and BE$ DT$ brainchild of NP.

↓ *Cropping (L=3)*

DT$ NN <SCH_TERM> BE$ owned by

An instance is further split into left and right sequences based on the position of <SCH_TERM>. We model the left and right sequences separately using soft pattern models. In the above example, "DT$ NN" is the left sequence and "BE$ owned by" is the right sequence.

Definition Sentence Retrieval: The definition sentence retrieval module identifies and extracts definition sentences from the relevant document set. Systems fielded at TREC rank definition sentences using two sets of features: definition patterns and bag-of-words pertinent to the target. Definition pattern matching is the most important feature used for identifying definitions. Xu *et al.* [19] and Hildebrandt *et al.* [6] employed various manually constructed definition patterns at both lexical and syntactic levels to identify specific linguistic constructs and assign different weights to the types of patterns. Other TREC systems [5, 20] also employ various manual definition patterns but they treat all patterns equally. All these systems use hard pattern matching. In contrast, soft matching patterns compute a probabilistic match for a test instance by combining individual slot and sequence probabilities. Cui *et al.* [2] experimentally demonstrated that soft pattern matching outperforms hard pattern matching in the definition sentence retrieval task. In this paper, we also adopt the soft pattern matching approach, but we augment the previous model with a rigorous mathematical foundation for soft pattern matching.

In addition to pattern matching, definitional QA systems also use the bag-of-words approach to rank extracted constructs. Blair-Goldensohn *et al.* [1] and Xu *et al.* [19] constructed a profile for the target by selecting centroid words, *i.e.,* words that frequently co-occur with the target. The profile is then used to rank definition sentences by their cosine similarity with the profile. Many systems also reinforce such profiles by exploiting word statistics from external resources, such as biographical web sites and encyclopedias. We follow the literature in constructing our system, by adopting the centroid method reinforced by existing definitions from biography.com and wikipedia.com. To retrieve definition sentences, we linearly combine the scores of pattern matching and bag-of-words ranking in the experiments [3].

Redundancy Removal: The redundancy removal module takes a list of ranked definition sentences as input. It produces the final answer by removing redundant sentences. A sentence is removed if its cosine similarity with any sentence already selected for the answer exceeds a predefined threshold [2].

2.1 Soft Pattern Matching

After preprocessing, each definition sentence is converted into a pattern instance that consists of left and right token sequences relative to the search target. At the training phase, token sequences are aligned and represented as a single vector *P:* $<S_1, S_2, \ldots, S_l>$ that combines information over all of the training sequences, where S_i represents the i^{th} slot (or position) left or right of the search target and contains tokens appearing in that position. Unlike hard matching patterns that generalize training instances into pattern rules, soft matching patterns model each token's distribution statistics in each slot over all training instances. Given a test sequence *T* with length l:$<t_1, t_2 \ldots\ldots t_l>$ where t_i is the token corresponding to the i^{th} slot, soft pattern models are used to model the generative probability of sequence *T*, given training sequences represented in vector *P*. In the next two sub-sections, we briefly review the background to our new soft pattern models before proposing the models in Section 3.

2.1.1 Language Modeling

Language modeling has been extensively studied in speech recognition, part-of-speech tagging and syntactic parsing [14]. N-gram language modeling is one important approach which models local sequential dependencies between adjacent tokens. Trigrams ($n=3$) are a common choice when large training corpora are available. We use a bigram ($n=2$) model in this paper, as we have a limited amount of training data. We also remedy problems with sparse data by smoothing n-gram probabilities. The first soft matching pattern model we introduce is based on n-gram language models, in which we incorporate linear interpolation [7] of unigrams and bigrams.

2.1.2 Hidden Markov Models in Information Extraction and Biological Sequence Modeling

Hidden Markov Models (HMMs) have been widely applied to speech recognition and various natural language processing applications [10], including information extraction (IE) and biological sequence modeling, which are most relevant to definition pattern matching.

IE relies heavily on patterns at the lexical, syntactic and semantic levels. Two types of extraction patterns are exploited in current IE systems: machine-induced hard matching rules [11] and probabilistic models such as HMMs. Skounakis *et al.* [15] applied hierarchical HMMs to the task of extracting binary relations in biomedical texts. They constructed two types of HMMs to represent words and phrases, which are two levels of emission units. Although these variations of HMMs also model pattern matching as token sequence generation, the topology they employ are more task specific and not general enough to be extended to other applications such as definition pattern matching.

HMMs recently have also been applied in computational biology to model protein families. Krogh *et al*. [8] utilized an HMM with a generic topology, called the Profile HMM (PHMM) (see Figure 2), to model multiple sequence alignment of protein families. PHMMs can be considered a probabilistic implementation of edit distance. It has different states for match, insertion and deletion operations. As it demonstrates flexible matching of lexico-syntactic patterns, we can easily adapt PHMMs for soft pattern matching by changing "amino acids" to "words and syntactic tags". The PHMM approach is our second model proposed.

3. GENERIC SOFT PATTERN MATCHING MODELS

In this section, we describe the derivation of our two soft pattern models and parameter estimation in detail.

3.1 Bigram Soft Pattern Model

We first adopt a bigram model to model pattern instances. While the original bigram model is simply a product of probabilities of all bigrams in a sequence, we apply linear interpolation of unigrams and bigrams to represent probability of bigrams. The reason is two-fold: (1) to smooth probability distribution to generate more accurate statistics for unseen data, and (2) to incorporate conditional probability of individual tokens appearing in specific slots. In particular, we model a sequence of pattern tokens as:

$$\begin{aligned} P(t_1 \ldots t_L) &= P(t_1 \mid \mu)\prod_{i=2}^{L}(\lambda P(t_i \mid t_{i-1}, \mu) + (1-\lambda)P(t_i, \mu)) \\ &= P(t_1 \mid S_1)\prod_{i=2}^{L}(\lambda P(t_i \mid t_{i-1}) + (1-\lambda)P(t_i \mid S_i)) \end{aligned} \tag{1}$$

where μ stands for the bigram model and $P(t_i|S_i)$ stands for the conditional probability of token t_i appearing in slot S_i. λ is the mixture weight combining the unigram and bigram probabilities. Note that we use the conditional probability of a unigram being in a slot to represent unigram probability. This is because the position of a token is important in modeling: for instance, a comma always appears in the first slot right of the target in an appositive expression. Incorporating individual slots' probabilities enables the bigram model to allow partial matching, which is a characteristic of soft pattern matching. In other words, even if some slots cannot be matched, the bigram model can still yield a high match score by combining those matched slots' unigram probabilities.

As test instances are often different in length, we normalize the log-likelihood of Equation (1) by the length *l* of the test instance:

$$P_{norm} = \frac{1}{l}(\log P(t_1 \mid S_1) + \sum_{i=2}^{l}\log(\lambda P(t_i \mid t_{i-1}) + (1-\lambda)P(t_i \mid S_i))) \tag{2}$$

where *l* denotes the number of tokens in the test instance.

Next, we estimate unigram and bigram probabilities by their maximum likelihood (ML) estimates:

$$P(t_i \mid S_i)_{ML} = \frac{|t_i(S_i)|}{\sum_k |t_k(S_i)|} \tag{3}$$

$$P(t_i \mid t_{i-1})_{ML} = \frac{|t_i(S_i) t_{i-1}(S_{i-1})|}{|t_i(S_i)|} \tag{4}$$

where $t_i(S_i)$ denotes that token t_i appears in slot S_i and $|t|$ denotes the frequency of the token t. In language modeling, ML estimates often suffer from the sparse data problem. It is even worse in our scenario because we count tokens with respect to slot positions, which makes the training data even sparser. As such, we need to employ some smoothing technique to counter the problem. For simplicity, we use Laplace smoothing on unigram probabilities (recall that bigram probabilities have already been smoothed by interpolation):

$$P(t_i \mid S_i) = \frac{|t_i(S_i)| + \delta}{\sum_k |t_k(S_j)| + \delta \times |N(t)|} \tag{5}$$

where $|N(t)|$ gives the total number of unique tokens in our training data and δ is a constant, which is 2 in our experiments.

Note that we count frequencies of words and general syntactic tags separately. General tags typically have a much higher frequency compared to individual words, and would thus skew the distribution if combined with words. Thus we need to separate the two types and estimate each token's unigram probability against its own set.

Estimating the mixture weight λ: We use the Expectation Maximization (EM) algorithm [4] to find optimal settings of λ. Specifically, we estimate λ by maximizing the likelihood of all training instances, given the bigram model:

$$\begin{aligned} \lambda &= \arg\max_{\lambda} \sum_{j=1}^{|INS|} P(t_1^{(j)} \ldots t_{l(j)}^{(j)} \mid \mu) \\ &= \arg\max_{\lambda} \sum_{j=1}^{|INS|} \frac{1}{l_j - 1} \sum_{i=2}^{l_j} \log(\lambda P(t_i^{(j)} \mid t_{i-1}^{(j)}) + (1-\lambda) P(t_i^{(j)} \mid S_i^{(j)})) \end{aligned} \tag{6}$$

$P(t_1|S_1)$ is ignored because it does not affect λ. λ can be estimated using the EM iterative procedure:

1. Initialize λ to a random estimate between 0 and 1, say 0.5.
2. Update λ using:

$$\lambda' = \frac{1}{|INS|} \times \sum_{j=1}^{f(INS)} \frac{1}{l_j - 1} \sum_{i=2}^{l_j} \frac{\lambda P(t_i^{(j)} \mid t_{i-1}^{(j)})}{\lambda P(t_i^{(j)} \mid t_{i-1}^{(j)}) + (1-\lambda) P(t_i^{(j)} \mid S_i^{(j)})} \tag{7}$$

where *INS* denotes all training instances and $|INS|$ is the number of training instances which is used as a normalization factor.

3. Repeat Step 2 until λ converges.

We set λ to 0.3 according to the experimental results.

3.2 PHMM Soft Pattern Model

Although the bigram model allows partial matching, it lacks the ability to deal with gaps in test instances. For instance, given training instances such as "<SCH_TERM> which is known for …", the trained bigram model cannot give reasonable match scores to test instances such as "<SCH_TERM> which is *best* known for …" or "<SCH_TERM> *, whose xxx* is known for …" even though they are simple variants of the training instances in which insertions or deletions occur. The gaps can be captured by PHMMs, which allow insertion and deletion operations in the matching process. Figure 2 shows the topology of a PHMM.

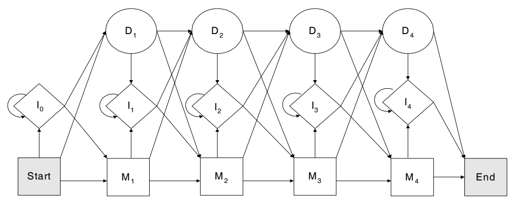

Figure 2. Illustration of the Profile HMM (*L=4*). Matching states are represented by M_i in squares, insertion states by I_i in diamonds, and deletion states by D_i in circles. Start and End states occupy the positions of 0 and *L+1*.

The PHMM contains a sequence of match states, which are denoted by M_i ($i=1..L$). These match states correspond to slots in pattern instances and determine model length L. Each match state can emit a token t from all tokens in the training instances with the emission probability $P(t|M_i)$. For each match state, there is a deletion state, denoted by D_i, which does not emit a token and is used to skip the corresponding match state. Insertion states emit a token t with the emission probability $P(t|I_i)$. Insertion states insert tokens after match or deletion states, as with the word "*best*" in the earlier example. While transitions from match states and deletion states always move forward in the model, insertion states allow self-loops, corresponding to multiple insertions. A token sequence representing a pattern instance can be generated by moving through this model with state transition probabilities $P(S_i|S_j)$. The deletion and insertion states allow the PHMM to model missing or unobserved words in training. Specifically, the probability of a sequence of tokens $t_1 \ldots t_N$ that are generated by moving through the states $S_0 \ldots S_{L+1}$ (the Start and End states are S_0 and S_{L+1}) is as follows:

$$\Pr ob(t_1 \ldots t_N \mid S_0 \ldots S_{L+1}, \mu) = T(S_{L+1} \mid S_L) \prod_{i=1}^{L} P(t_{n(i)} \mid S_i) T(S_i \mid S_{i-1}) \tag{8}$$

where μ stands for the model. $P(t_{n(i)}|S_i)$ is set to 1 when S_i is a deletion state. To recognize a definition pattern, we choose the most probable state path in the above equation to approximate the probability of the sequence being given all possible state paths. The rationale is that most often, the most probable state path gives a much higher probability than any other paths. Equation (8) can be efficiently calculated by the forward-backward algorithm [10]. We employ the Viterbi algorithm [10] to find the most probable state path. In Figure 3, we show an example to illustrate how the PHMM finds the optimal path to account for the "gaps" between training instances and the test instance. Although the training data does not contain any instance that has "known" in Slot 1 and "NNP" in Slot 4, the PHMM automatically selects the path that goes through a deletion state to skip Slot 1 and uses an insertion state to emit "NNP". Thus, the tokens are re-aligned with their most probable occurring slots such that the unseen test instance can still obtain a reasonable generative probability.

Estimation of the model: During training, we need to estimate transition and emission probabilities for the PHMM. The training process, also called the estimation process, can be accomplished by employing the standard Baum-Welch algorithm [10]. Corresponding to our adaptation to the calculation of sequence probability, we use the Viterbi algorithm to determine the path with the highest probability during the re-estimation process, unlike the standard Baum-Welch algorithm which considers all possible paths which are weighted by their probabilities.

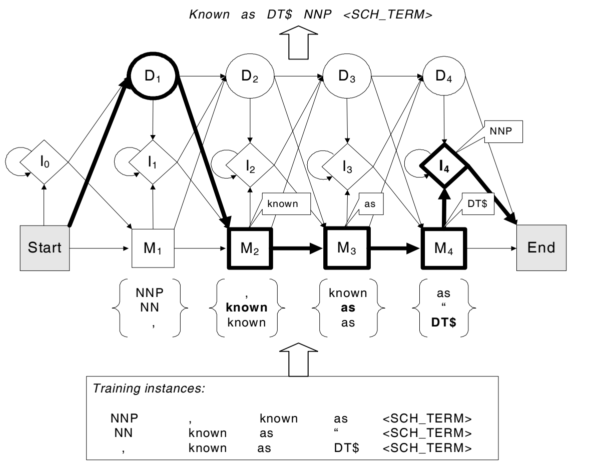

Figure 3. Generating a test instance with gaps using the PHMM. The optimal path is in bold, and words or tags emitted are shown in callouts.

Initialization of the model: Although probabilities in a PHMM can be estimated automatically using an iterative EM algorithm starting with random or uniform probabilities, the re-estimation process can only guarantee that the model reaches local maxima. In addition, in capturing definition patterns, definition expressions are diverse and sparse in terms of both lexical tokens and POS tags. If we start with random or uniform setting of the model, we would likely end up with an unsatisfactory model that gives close estimates of different possibilities. To make training manageable given our small training set, we assume that the most probable state path for a sequence should go through as many match states as possible. The reason is that although insertion and deletion states add flexibility, they may hurt generalization of underlying definition patterns if the model gives high probabilities of going through them. Specifically, we set the emission probabilities for each match and insertion state using the smoothed maximum likelihood estimate of the emission probabilities (Equation 5). We adjust the value of δ such that the probability of emitting a token from match states is higher than that of insertion states. We set the initial state transition probabilities to the inverse proportion of the number of transition links from a state.

3.3 Combining Left and Right Sequences

The pattern matching score for a test instance against definition patterns is obtained by combining the soft pattern matching scores for both the left and right sequences in the context of the search target. We use the linear combination of the scores:

$$P(ins \mid C) = \alpha P(left\_seq \mid SP_L) + (1-\alpha) P(right\_seq \mid SP_R) \qquad (9)$$

where *ins* represents a test instance and *C* denotes the context model. $P(left\_seq|SP_L)$ and $P(right\_seq|SP_R)$ give the probabilistic pattern matching scores of the left and right sequences of the instance, given the corresponding soft pattern (SP) matching models. α is the mixture weight. We adopt an EM algorithm similar to that in Section 3.1 to estimate the value of α. We set α to 0.3 in our experiments.

3.4 Discussions on the Two Generic Soft Matching Models

We have presented two generic soft pattern models: Bigram SP and PHMM SP. These two models differ in their complexity. The bigram model can be considered a simplified first-order Markov model with one state for each token. It directly captures sequence information using bigram probabilities. In contrast, PHMM has a more complicated topology that aggregates token sequence probability into state transition probabilities. Theoretically, the PHMM needs more training data to converge to an accurate model as it has more parameters to estimate. Appropriate initialization (as shown in Section 3.2) for the PHMM also aids convergence to the global maxima. An advantage is that PHMMs are less sensitive to model settings, *e.g.,* model length, because it makes its transitions between hidden states which correspond to aggregations of tokens rather than directly between surface tokens. We present experiments in Section 4 to validate these conjectures.

Despite their differences, the two models are inherently connected because both the models deem definition patterns as sequences of tokens. They model the same structural information for patterns: First, both models capture the importance of a token's position in the context of a search target: the bigram model uses unigram probabilities while the PHMM model uses emission probabilities to represent a token's independent probability of appearing in each position. Second, both models account for the sequential order of tokens. This sequential information is captured by bigram probability in the bigram model and state transition probability in the PHMM model. Thus, the soft matching method proposed in Cui *et al.* [2] may be considered a special case of our Bigram SP model.

Our assumption is that all definition pattern instances embedded in definition sentences are generated by a single model. Although it may be advantageous if we could train separate probabilistic models for different types of definition patterns, limited training prevents this, and such models impair the objective of establishing a uniform matching model for definition patterns.

4. EVALUATIONS

We have three goals in our evaluations: (1) to compare the performance of our soft matching models against other state-of-the-art pattern matching techniques in the context of a standard definitional QA system, and to study the two models' sensitivity to: (2) model length, and (3) amount of training data.

4.1 Evaluation Setup

4.1.1 Data Set

We employ the data set from the TREC-13 Question Answering Task. It includes the AQUAINT corpus of over one million news articles and 64 definition question[2] and answer pairs. We use the data set from the TREC-12 definitional QA task as training data, which shares the same corpus with TREC-13 and includes an additional 50 definition question and answer pairs. Based on the answer nuggets (ground truth, manually labeled data) provided by TREC for these 50 questions, we manually label all sentences that cover the nuggets from the corpus as definition sentences. In total,

[2] The test data for TREC-13 includes 65 definition questions. NIST drops one in the official evaluation.

we obtain 761 labeled definition sentences as training data for estimating the soft matching models.

4.1.2 *Evaluation Metrics*

We adopt the evaluation metrics used in the TREC definitional question answering task [16, 17]. TREC provides a list of essential and acceptable nuggets for answering each question. We use these nuggets to assess the various QA systems in our evaluation in both manual and automatic assessments. In the manual assessment used in official TREC evaluations, an assessor examines how many essential and acceptable nuggets are covered in the returned answer. Each definition is scored using nugget recall (NR) and an approximation to nugget precision (NP) based on answer length. These scores are combined using the F_3 measure with recall being weighted three times as important as precision [17].

In addition to manual assessment, we perform automated evaluation using ROUGE [9]. Automatic scores can be a good supplement to manual evaluations for two reasons. First, as Xu *et al.* [19] suggested, ROUGE gives automatic scores that are highly correlated with manual counting of nuggets. Second, the manual checking of nuggets is subject to inconsistent scoring across runs [16]. ROUGE is a metric originally designed for summarization evaluation and has previously been adapted for definitional QA evaluation [19]. We use the metric ROUGE-3, which counts the trigrams shared between the official answer and the system answer.

To perform automatic scoring, for each search target, we construct five groups of sentences as the gold standard. According to TREC-13 guidelines, gold standard sentences are selected based on answers to the factoid/list questions about the target and "other" information about the target, which includes essential and acceptable nuggets. We give details of how to construct the gold standard in the Appendix. We use two ROUGE metrics: ROUGE-3-ALL (R3A) for evaluations against all sentences in the gold standard and ROUGE-3-ESSENTIAL (R3E) for evaluations against those sentences that contain only factoid/list answers and essential nuggets in the gold standard list. The final ROUGE scores are the average scores obtained by running the evaluation tool over the five groups of gold standard lists.

4.1.3 *Comparison Systems*

In our experiments, the base definition generation system used is the system discussed in Section 2 and illustrated in Figure 1. In evaluations, we only vary the definition pattern matching module while holding constant all other components and their parameters. For comparison, we apply a set of manually constructed hard matching definition patterns which has demonstrated state-of-the-art performance as the baseline system. The patterns combine those used in Cui *et al.* [2] and Hildebrandt *et al.* [6], which comprise the most complete published list of patterns to our knowledge. In particular, we use the following comparison systems:

(1) HP-Filter: This system employs the method used in Xu *et al.* [19] and Hildebrandt *et al.* [6] where bag-of-words is used to rank those constructs matched by any manual definition pattern.

(2) Original SP: We also use the soft pattern matching method proposed in Cui *et al.* [2], and adopt the same parameter settings.

In our evaluations, we set answer length *N* to 14 sentences for all systems to approximate the desirable answer length used in successful TREC systems [19, 6, 2].

4.2 Performance Evaluation

In the first evaluation, we assess the performance obtained by the two soft matching models against that of the comparison systems. We set model length *L* to optimal values based on experiments which we will present in the next subsection. We list the evaluation results in Table 1. We take HP-Filter as the baseline system for comparison with the soft pattern matching models.

Table 1. F_3 performance comparison (percentage improvement shown in brackets; ** and * represent different significance levels by t-test: $p \leq 0.01$ and 0.05, respectively)

Configurations	HP-Filter (Baseline)	Original SP	Bigram SP	PHMM SP
R3A	0.2106	0.2233 (+6.00%)	0.2303 (+9.37%)	0.2234 (+6.08%)
R3E	0.2286	0.2378 (+4.00%)	0.2553 (+11.67%)*	0.2496 (+9.18%)
NR	0.5027	0.5376	0.5519	0.5420
NP	0.3159	0.3238	0.3403	0.3264
F_3	0.4633	0.4937 (+6.56%)**	0.5088 (+9.83%)**	0.4971 (+7.30%)**
Correlation F_3, R3A	0.63	0.63	0.66	0.61
Correlation F_3, R3E	0.63	0.69	0.67	0.60

From Table 1, we arrive at the following:

1. We reaffirm the conclusion drawn by Cui *et al.* [2] that soft matching patterns outperform manually constructed hard matching patterns in both manual and automatic evaluations. With the manual F_3 measure, all three soft pattern models perform significantly better than the baseline ($p \leq 0.01$). The significance measures change slightly when ROUGE scores are used. With R3E, only the bigram and PHMM models achieve significant improvement over the baseline (*p=0.03* and *p=0.08*). The original and PHMM models achieve similar performance in R3A scores while the bigram model achieves some improvement. We conjecture that the differences among the significance tests are due to the long standard answers. Recall that we have compiled a list of sentences as the gold standard for ROUGE evaluation. While the human assessor is able to figure out the real answer nuggets embedded in the system-returned answers, the ROUGE evaluation tool is likely to overestimate recall due to the long standard answers.

2. We note that both Bigram SP and PHMM SP outperform Original SP in all scores. Bigram SP outperforms Original SP by 7.36% (*p=0.09*) and 3.05% (*p=0.1*) in R3E and F_3 scores, respectively. PHMM SP achieves 5.00% improvement over Original SP in R3E scores. These results show that the preliminary soft matching method is not optimized in parameter setting. Finding best parameters is often tedious and difficult for such ad hoc systems. In contrast, Bigram SP and PHMM SP provide a sound framework for parameter estimation. This should

facilitate the migration of the two generic soft matching models to other applications.

3. We observe that the manual F_3 scores are highly correlated with the automatic metrics R3A and R3E despite that the ROUGE scores might have minor disturbance due to the long gold standard answers. We calculate statistical correlation between F_3 and ROUGE scores. Correlation measures vary from -1 (perfectly anti-correlated) to 1 (perfectly correlated). All the correlation measures in our evaluations are between 0.6 and 0.7, which indicate strong correlations between the metrics.

4.3 Analysis of Sensitivity to Model Length

As the parameters of the models are estimated automatically, we vary the only arbitrarily set factor – model length – for the two models. We list the ROUGE scores for both models when varying their model length (number of slots) from 2 to 6 in Table 2.

In Table 2, we see that Bigram SP obtains the best performance with the model length of 3 while PHMM SP achieves the highest performance with the model length of 4. Both models slacken in their performance when more slots are used.

We also compare percentage change in performance against the highest score for each scoring metric. The performance of the bigram model fluctuates more over different model lengths compared to PHMM SP. This is evidence that PHMM SP may be more stable amid changes in model typology.

Another observation is that with model lengths of 5 and 6, PHMM SP performs better than the bigram model. We hypothesize that the PHMM model may be more capable of dealing with longer contexts.

Table 2. Performance with different model lengths. The percentage values in parentheses are difference measures compared to the maximum. Note that PHMM SP's minimum length for training is 3.

Model Length (# Slots)	2	3	4	5	6
PHMM SP R3A	N/A	0.2139 (-4.25%)	0.2234	0.2190 (-1.97%)	0.2125 (-4.88%)
PHMM SP R3E	N/A	0.2369 (-5.09%)	0.2496	0.2422 (-2.97%)	0.2367 (-5.17%)
BIGRAM SP R3A	0.2128 (-7.60%)	0.2303	0.2165 (-6.00%)	0.2152 (-6.56%)	0.2086 (-9.42%)
BIGRAM SP R3E	0.2340 (-8.34%)	0.2553	0.2363 (-7.44%)	0.2354 (-7.80%)	0.2346 (-8.11%)

4.4 Analysis of Sensitivity to Amount of Training Data

In this evaluation, we experiment with different amounts of training data. We divide the training data into two or three equal portions. We train the bigram and PHMM models by using different amounts of training data while testing on the same test data set as before. We perform multiple runs with different portions of training data, and average the scores obtained by the system. Table 3 lists the results.

Table 3. Performance comparison across varying amounts of training data.

	Training Data size (fraction of whole training corpus)		
	1/3	1/2	1
PHMM R3A	0.2110	0.2179 (+3.24%)	0.2234 (+5.85%)
PHMM R3E	0.2311	0.2402 (+3.93%)	0.2496 (+8.00%)
Bigram R3A	0.2229	0.2269 (+1.76%)	0.2303 (+3.32%)
Bigram R3E	0.2478	0.2510 (+1.29%)	0.2553 (+3.03%)

Table 3 shows that PHMM SP achieves higher improvement than Bigram SP when more training data is used. On the other hand, comparing the performance difference between the PHMM and bigram models with different amounts of training data reveals that with more training data, the performance difference between the two models narrows. For instance, the difference decreases from 7.22% to 2.28% and from 5.61% to 3.09% in R3E and R3A scores respectively when we change from using one third to using the full amount of training data. This observation supports our conjecture that PHMM requires a larger amount of training data for parameter estimation. Although Table 1 shows the bigram model performing better, we believe that with enough training data, PHMM SP may outperform the bigram model.

5. CONCLUSIONS AND FUTURE WORK

We have proposed two generic soft pattern models: one based on a bigram language model and the other on the PHMM. Both provide formal probabilistic methods to model lexico-syntactic patterns represented by token sequences. In particular, we have shown that PHMM overcomes the problem of gaps caused by language variations in pattern matching. Our experiments show both models obtaining significantly better performance than carefully constructed hard matching patterns in a definitional QA system. Although the bigram model shows slightly better performance between the two new models in our evaluations, we believe that the PHMM model can perform better with more training data. Moreover, as the PHMM model has shown to be more tolerant to language variations, it is likely to be suitable in applications with diverse training and test instances.

Providing formal models for modeling contextual lexico-syntactic patterns is the main contribution of this work. Our two soft matching models are generic and can be extended to related areas that require modeling of contextual patterns, such as information extraction (IE). The pattern matching problem in IE tasks are formally the same as definition sentence retrieval. When conducted on free texts, an IE system can also suffer from various unseen instances not being matched by trained patterns. Xiao *et al.* [18] have demonstrated that soft pattern matching greatly improves recall in an IE system. Although some HMM topologies have been employed for IE tasks, our models are more generic and require less configuration and parameter tuning with changing domains. The models can help IE systems overcome difficulties caused by language variations in pattern matching.

6. ACKNOWLEDGMENTS

The authors are grateful to Wee Sun Lee for enlightening us on Profile HMM and for his valuable suggestions on the draft of the paper. We also thank Alexia Leong for proofreading the paper. Thanks also go to anonymous reviewers whose comments have helped improve the final version of this paper. The first author is supported by the Singapore Millennium Foundation Scholarship in his PhD studies (ref no. 2003-SMS-0230).

7. REFERENCES

[1] S. Blair-Goldensohn, K.R. McKeown and A. Hazen Schlaikjer, *A Hybrid Approach for QA Track Definitional Questions*, Proc. of TREC 2003, 2003, pp. 336-343.

[2] H. Cui, M.-Y. Kan and T.-S. Chua, *Unsupervised Learning of Soft Patterns for Generating Definitions from Online News,* Proc. of WWW '04, New York, 2004, pp. 90-99.

[3] H. Cui, M.-Y. Kan, T.-S. Chua and J. Xiao, *A Comparative Study on Sentence Retrieval for Definitional Question Answering*, SIGIR Workshop on Information Retrieval for Question Answering (IR4QA), Sheffield, U.K., 2004.

[4] A.P. Dempster, N.M. Laird and D. B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, 39:1-38, 1977.

[5] S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams and J. Bensley, *Answer Mining by Combining Extraction Techniques with Abductive Reasoning*, Proc. of TREC 2003, 2003.

[6] W. Hildebrandt, B. Katz and J. Lin, *Answering Definition Questions with Multiple Knowledge Sources*, Proc. of HLT/NAACL 2004, Boston, MA, 2004, pp. 49-56.

[7] F. Jelinek and R. L. Mercer, *Interpolated estimation of markov source parameters from sparse data*, Proc. of the Workshop Pattern Recognition in Practice, Amsterdam, Holland, 1980, pp. 381-397.

[8] A. Krogh, M. Brown, I.S. Mian K. Sjolander and D. Haussler, *Hidden Markov Models in Computational Biology - Applications to Protein Modeling,* J. Mol. Biol. (1994) 235, pp. 1501-1531.

[9] C.-Y. Lin and E.H. Hovy, *Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics*, Proc. of HLT-NAACL '03, Edmonton, Canada, 2003, pp. 71-78.

[10] C.D. Manning and H. Schtze, editors. *Foundations of Statistical Natural Language Processing*, The MIT Press, Cambridge, MA, 1999.

[11] I. Muslea, *Extraction patterns for information extraction tasks: A survey,* Proc. of AAAI-99 Workshop on Machine Learning for Information Extraction, 1999, pp.1-6.

[12] D. Ravichandran and E. Hovy, *Learning Surface Text Patterns for a Question Answering System*, Proc. of ACL '02, Philadelphia, July 2002, pp. 41-47.

[13] E. Riloff, and J. Wiebe, *Learning Extraction Patterns for Subjective Expressions*, Proc. of EMNLP '03, 2003.

[14] R. Rosenfeld, *Two decades of statistical language modeling: Where do we go from here*, Proc. of the IEEE, 88, August, 2000, pp. 1270-1278.

[15] M. Skounakis, M. Craven, and S. Ray, *Hierarchical hidden markov models for information extraction*, Proc. of IJCAI '03, 2003.

[16] E.M.Voorhees, *Overview of the TREC 2003 question answering track*, Proc. of TREC 2003, 2003.

[17] E.M. Voorhees, *Overview of the TREC 2004 question answering track,* Proc. of TREC 2004, 2004.

[18] J. Xiao, T.-S. Chua and H. Cui, *Cascading Use of Soft and Hard Matching Pattern Rules for Weakly Supervised Information Extraction*, Proc. of COLING '04, Geneva, Switzerland, 2004, pp.542-548.

[19] J. Xu, R. M. Weischedel and A. Licuanan, *Evaluation of an extraction-based approach to answering definitional questions*, Proc. of SIGIR '04, Sheffield, UK, 2004, pp. 418-424.

[20] H. Yang, H. Cui, M.-Y. Kan, M. Maslennikov, L. Qiu and T.-S. Chua, *QUALIFIER in TREC 12 QA Main Task*, Proc. of TREC 2003, 2003, pp. 54-63.

8. APPENDIX

According to TREC-13 QA guidelines (Voorhees 2004), definitional QA systems are required to present "other" information about the search target that is not covered by the factoid or list questions related to the target. As our purpose is to evaluate a definitional QA system, we perform the following alterations to make the evaluation complete: We use a list of answer patterns for the factoid and list questions about a target to search for sentences that contain the answers. We treat these factoid/list answers as essential nuggets and add the answer sentences to the gold standard list. This is based on the guideline that factoid/list questions are about the most essential information about the target. We choose sentences because answers for factoid/list questions are only phrases and other nuggets about the target are often ungrammatical text fragments. The original form of answers and nuggets cannot be matched by ROUGE in most cases. As the same answer may be embedded in different sentences, we search for up to five sentences for each factoid/list answer and for each nugget in the definition part. Accordingly, we create five groups of gold standard lists for each target. Besides factoid and list answers, we also add sentences containing essential and acceptable nuggets to "other" questions to the gold standard list.