Chapter 1
Introduction

1.1 Why Automatic Lexicon Acquisition

A large-scale electronic dictionary is the fundamental component to many natural language processing (NLP) applications such as spelling correction, grammar checking, text-speech conversion, intelligent Chinese input methods and machine translation. However, *new words* (or *unknown words*, as defined in [Wang 95]) and new *compound words*, both for English and Chinese (and, of course, other languages), are increasing rapidly in various domains, especially after the blooming growth of the Internet community.

Including such lexical entries in the system dictionary beforehand normally improves the performance of an NLP system significantly, because it reduces the requirement to analyze these lexical entries. In an English-Chinese machine translation system, for instance, the translation quality will be greatly improved if the English unknown compounds are identified and included before the translation process begins. On the other hand, if a compound is not in the dictionary, it might be translated incorrectly because many compounds are not *compositional*, which means that the translation of a compound is not the composite of the respective translations of the individual words [Chen 88]. For example, the Chinese translation of 'green house' is not the composite of the Chinese translations of 'green' and 'house'. Without registering such compounds in the dictionary, the translation quality of the target text will be greatly degraded. Furthermore, the number of parsing *ambiguities* will also increase due to the large number of possible parts of speech combinations for the individual words if such new compounds are unregistered. It will then reduce the *accuracy* rate in disambiguation, degrade the processing or translation *quality* and increase the processing *time*.

In addition, for an NLP task such as machine translation, a computer-translated manual is usually concurrently processed by several posteditors in practical operations. Therefore, maintaining the consistency of the translated terminologies among different post-editors is very important, because terminological consistency is one of the major advantages of machine translation over human translation. If all the terminologies can be entered into the dictionary beforehand, the consistency can be automatically maintained, the translation quality can be greatly improved, and lots of post-editing time and *consistency* maintenance cost can be saved.

In analyzing Chinese text, it is also known that the results of Chinese word segmentation, which is used to identify the word boundaries among the Chinese characters, will not be satisfactory if a large number of unknown words or compounds are not registered in the system dictionary. Such bad segmentation will thus hinder further processing of the Chinese text, including the above mentioned problems for a machine translation system. Quickly acquiring *compound words* and *new words* that are not registered in an existing dictionary is thus very important.

Since new compounds and new words are usually produced through some very productive word formation rules, it is impossible to exhaustively store all such lexical entries in a dictionary. Furthermore, from the experiences in operating a machine translation system [Chen 91], it was found that the number of new lexical entries may exceed several thousands, especially for large translation projects. Under such circumstances, human involvement will be costly and time-consuming, if all new words or compounds are acquired by manually scanning the corpus. In addition, the lexicographers usually have difficulty to judge objectively what new terms should be included into the lexicon if certain quantitative indices are not provided. Therefore, without an automatic method for lexicon acquisition, the dictionary may not be adapted promptly to the quick changing world, with little cost and high coverage in different domains. Therefore, spotting and updating such terminologies beforehand without much human effort is important; an *automatic* and quantitative tool for lexicon acquisition from the text corpus is thus seriously required.

1.2 Lexicon Extraction Tasks

In this dissertation, two typical lexicon extraction tasks, namely English compound word extraction and Chinese unknown word identification, will be studied in order to explore performance optimization techniques in these two different languages. In the English compound word extraction task, we are given an input text file containing word streams like ‘... please consult Microsoft Words User Manuals for more details ...’, and the main task is to identify whether compound word candidates, such as ‘please consult’, ‘consult Microsoft’, ‘Microsoft Words’, ‘Words User’, ‘User Manuals’, etc., are qualified as real compound words. The identification is conducted by a filter or classifier based on certain features associated with the n-grams (i.e., n-word groups, such as ‘Words User’ and ‘User Manuals’); and a training corpus is available, from which statistical knowledge about compound and non-compound n-grams is acquired. In this task, all the compound words in the training corpus are identified in advance. Therefore, the training process for this task is supervised.

The performance of the lexicon extraction tasks, like most information retrieval tasks, is evaluated with the *precision rate* and the *recall rate* of the system, where precision is the percentage of real compounds in the extracted output word list, and recall is the percentage of real compounds in the entire input corpus that are correctly extracted to the output word list. More generally, an objective function of precision and recall, such as the average of precision and recall or the F-measure [Appelt 93, Hirschman 95, Hobbs 96], is used to evaluate the joint precision-recall performance (Section 2.1). Our focus is therefore to develop optimization techniques for maximizing such objective functions.

In the Chinese unknown word extraction task, the unknown words (i.e., words which are not registered in the system dictionary) within a Chinese text corpus are to be identified, assuming that we have a system dictionary containing some known words. This task is more complicated than the English compound word extraction task, since the Chinese characters are not delimitted by any spaces, and thus the word boundaries are not known. For instance, the Chinese input sentences may contain character streams like ‘移送台中少年法庭審理’ (‘send ... to the Taichung Youth-Court for investigation’), where there is no space between the words ‘移送’, ‘台中’, ‘少年法庭’ and ‘審理’. Under this situation, a segmentation module might be necessary to identify the word boundaries, and may produce segments like: ‘移 送 台中 少年 法庭 審 理’ (which is actually not the most preferred segmentation pattern since there might have unknown words like ‘移送’, ‘少年法庭’ and

‘審理’). Once a segmentation is acquired, it is then necessary to check whether substrings like ‘移送’, ’移送台中’, ‘送台中’, ’台中少年’, ’少年法庭’, etc., are qualified new words with a filter.

Note that, since many auxiliary tools, such as part-of-speech taggers, for annotating the Chinese text corpora with extra information are not yet well developed, we will assume that the only resource is the input text corpus and a system dictionary (which contains some known words); therefore, the statistical knowledge about words and non-words cannot be acquired from the corpus directly. The training procedure for this task is therefore essentially unsupervised.

There are several problems with conventional lexicon extraction approaches for the above two tasks. The following sections briefly identify such issues and propose our strategies to overcome the identified problems. The theoretical frameworks and implementation details are left to the other chapters.

1.3 Problems with English Compound Word Extraction and General Strategies

The English compound word extraction problem can be modeled as a two-class classification problem, in which potential compound candidates (such as ‘Words User’ and ‘User Manuals’) are firstly collected from the input corpus and then classified into either the compound class or the non-compound class. Many related issues on such an extraction task, either rule-based or corpus-based, had been addressed in the literature for extracting English or Chinese terminologies [Church 90, Calzolari 90, Bourigault 92, Wu 93, Smadja 93, Su 94b, Tung 94, Chang 95, Wang 95, Smadja 96]. Our focus will be on statistical methods for English compound word extraction, since statistical approaches have many advantages for large-scale systems in automatic training, domain adaptation, systematic improvement, and low maintenance cost.

1.3.1 Problems with Conventional Filtering Approaches

Most statistical approaches [Church 90, Smadja 93, Tung 94, Wang 95, Smadja 96] for terminology extraction rely on word association metrics, such as frequency [Wang 95, Smadja 96], mutual information [Church 90], dice metrics [Smadja 93] and entropy [Tung 94] to identify whether a group of words is a potential compound (or highly associated collocate). (Refer to Chapter 2 for the definitions of some of the features.) The mechanisms for applying such features are often based on simple filtering-and-thresholding statistical tests; a compound candidate will be filtered out (or classified as non-compound) if its association metric is below a threshold; when multiple features are available, the features are usually applied one-by-one independently with different heuristically determined thresholds. Such approaches can be implemented easily, and encouraging results were reported in various works. However, there are several technical problems with such filtering approaches.

First of all, most simple word association features, such as frequency and mutual information, can only indicate whether an n-gram (i.e., a group of n words) is highly associated; however, high association does not always implies that it is a compound, since there are other syntactic (and even semantic) constraints which will also produce highly associated n-grams. For instance, the word pair "is a" has sufficiently high frequency of occurrence and high mutual information. Nevertheless, it is not a compound word since such a construct is produced due to syntactic reasons. Many long collocates extracted by such filtering methods are also of this category [Smadja 96]. Therefore, many highly associated non-compound n-grams might be mis-recognized as compounds.

Although it is known that *syntactic information* is useful in resolving such problems, there are few works for integrating high level syntactic or semantic features, such as parts of speech, with known word association metrics in a *simple* and effective way for the extraction tasks. A part of speech related metric is therefore proposed in this dissertation to formulate the syntactic constraints among the constituents of potential compound candidates. Such integration between word association metrics and syntactic constraints in a uniform formulation is important, since syntactic constraints are more closely related to the generation of the unregistered compounds, and it is desirable to apply simple statistical tests based on such features, instead of using complicated syntactic processing.

Second, since the association features are often applied independently in different stages for filtering when multiple features are available, it does not take advantages of jointly using all discrimination information at the same time to acquire the best system performance. For instance, by filtering out low frequency candidates and then filtering out candidates with low mutual information, we may filter out low frequency candidates which actually have high mutual information. If the filtering mechanism is based on *both* frequency and mutual information jointly, the system performance is expected to be better.

In fact, it is well known that the performance is usually improved if multiple features are jointly considered, instead of using a single feature or applying multiple features independently. Therefore, what is really important is an approach which could combine all available features for acquiring the best performance in the extraction task.

However, several factors must be carefully considered in order to enjoy the discrimination information provided by multiple association features. For instance, many features proposed in the literatures are highly correlated. Therefore, the correlations among the association features must be included into the statistical model in order to acquire the best achievable performance. In this work, we will therefore use (a mixture of) multivariate Gaussian density functions to incorporate the effects of the inter-feature correlation. Furthermore, it is desirable to use only the most discriminative features and reject features that are either non-discriminative or redundant with respect to other more discriminative features when combining the features. In this dissertation we therefore propose an integrated method, which select the most appropriate features automatically, for combining a set of useful features. In particular, optimization based on frequency, mutual information, dice metric, contextual entropy and parts of speech information will be surveyed.

Furthermore, many useful features, such as part of speech information, are not continuous-valued. Integration of such features with continuous-valued features is thus also interesting in case such features are available. We therefore also propose some techniques to deal with the integration of such kinds of features.

Finally, the most important performance indices for terminology extraction and information retrieval tasks are the precision rate and recall rate. Most previous works, however, do not have explicit performance indices or *optimization criteria* to optimize in designing the filter or classifier. Normally, a heuristically determined threshold is used for each association feature so that n-grams whose word association metric is below a particular threshold are rejected as new terminologies. When precision is emphasized, a high threshold is adopted, without regarding the degradation of the recall performance; when recall is of upmost important, a low threshold is given, without considering the effect on the precision performance.

1.3.2 Strategies for Precision-Recall Maximization

Since the thresholds in conventional filtering approaches are not automatically determined based on any optimization criterion of the precision-recall performance, it is difficult to decide what values are to be used in a particular application. Hence, it is unable to achieve the best performance in terms of an object criterion function of precision and recall (such as the average of precision and recall).

From a user's point of view, it is also very inconvenient to use such tools since the user never know what thresholds to use in a particular task. In fact, most users would prefer to specify a *criterion* of the system performance rather than the *thresholds* to achieve such a criterion. Therefore, it is desirable to develop a mechanism which could optimize the system performance according to users' preference over the precision-*vs*-recall profile, rather than developing a tool which relies on the users to adjust the thresholds by trial-and-error. In other words, it is desirable to maximize an objective function of the *precision* and *recall* rates using an automatic method.

Unfortunately, there is no simple analytical methods, in the literatures, which are capable of achieving maximum precision-recall classification in terms of any user specified objective function of precision and recall. Therefore, a two-stage optimization scheme is proposed here as an approximation to the optimal classifier for achieving the best joint precision-recall performance.

In the first optimization stage, an optimal classifier, whose optimization criterion is closely related to good joint precision-recall performance, is used as the base classifier for extracting compounds. Currently, a minimum error classifier [Devijver 82, Juang 92], whose expected classification error is minimal, is adopted to minimize the classification error for compound extraction.

The minimum error classifier is used here for the following reasons. (1) The discrimination function, which is used to classify the potential compound candidates into compound and non-compound, has a simple analytic form. The implementation of such an optimal classifier is therefore easy. (2) Reducing classification error, in general, will improve precision and recall, and hence improve most joint precision-recall performance indices. (See Section 2.3 for quantitative explanation.)

In the second optimization stage, a non-linear learning method is used to adjust the parameter values of the minimum error classifier in order to achieve maximum precision-recall performance, starting from the minimum error status. There are two reasons for doing this. (1) The minimum error classifier may not really achieve minimum errror due to imperfect modeling or estimation errors in estimating the parameter values. For instance, using a simplified probability density function may introduce modeling error, and using a maximum likelihood estimator may not really achieve minimum classification error [Su 94a]. Adjusting the parameters is thus required for further improving the system performance. (2) The minimum error rate criterion is not equivalent to the maximum precision-recall criterion (even though the minimum error performance is reached.) For instance, the same error rate may result from different types of classification errors (i.e., misclassifying compounds into non-compounds or misclassifying non-compounds into compounds), and thus may correspond to different precision-recall preformance. Therefore, it is possible to tune the system for the best joint precision-recall performance even if the minimum error performance is reached.

A non-linear learning approach which could optimize an objective function of the precision and recall performance is therefore proposed in the current work to adjust the parameters of the minimum error classifier. In particular, optimization for maximum *weighted precision & recall* (WPR) and maximum F-measure (FM) are surveyed in detail. With such a device, a lexicographer can simply specify a preferred precision-*vs*-recall profile for the extraction task, and let the learning algorithm adjust the parameters to meet the optimization criterion, instead of trying different thresholds empirically.

1.4 Problems with Chinese Unknown Word Extraction and General Strategies

1.4.1 General Scheme in Previous Approaches

In the Chinese unknown word extraction task, we are given a large text corpus and a system dictionary with some known words, where there is no delimiter (like spaces) between words in the corpus; the major goal is to extract new words not registered in the system dictionary in a way that reaches maximum precision-recall performance.

A few closely related works had been introduced for finding Chinese new words (or unknown words) from a large corpus [Chiang 92, Lin 93a, Lin 93b, Tung 94, Chang 95, Wang 95]. Some of the above works for Chinese unknown word extraction [Chiang 92, Lin 93a, 93b, Tung 94] need a pre-annotated corpus for supervised training. For instance, [Tung 94] uses a segmented corpus with part-of-speech tags for training parameter values. Since human cost is high in preparing such a training corpus, especially for Chinese text corpora whose part of speech tagger is still not well developed with high accuracy, we will thus focus on the unsupervised method for new word extraction. Since the word boundaries are not known, a segmentation module is usually used, in traditional approaches, to form potential candidates before submitting the candidates to a filter or classifier.

Although not exactly the same, many previous works, such as [Tung 94, Wang 95], can be roughly characterized by the following segmentation-merging-filtering-and-disambiguation steps: (1) The Chinese text corpus is *segmented* based on an existing dictionary (hereafter, the *system dictionary*) using a rule-based approach or stochastic word segmentation model (whose model parameters are estimated from a large manually segmented corpus). (2) Since many *unknown words* will be segmented into known words and single characters after segmentation, potential unknown word candidates are formed by *merging* adjacent segments (i.e., known words or single characters) in the segmented corpus. (3) Afterward, a set of association metrics or testing statistics, such as mutual information [Church 90], entropy [Tung 94], or Xtract-style strength measure [Smadja 93, Wang 95] are used by a *filter* to filter out candidates which have low associations. (4) Finally, an optional step is used to resolve *ambiguity* between overlapped candidates which are identified as new words at the same time [Tung 94]. For instance, '漁業 區 附近' will produce the overlapping candidates: '漁業區' and '區附近', which need be resolved to identify who is a stronger competitor. In [Tung 94], the entropy information, which was also used by the filter, is used in a different way to tell which overlapping candidate is the stronger one. Furthermore, the above 4 steps are usually executed only once without iteratively looping back.

The above-mentioned non-iterative segmentation-merging-filtering-disambiguation approaches can easily be implemented to acquire new word candidates for human post-editing; by adjusting some thresholds *via* trial-and-error, the precision or recall can also be adjusted to fit the lexicographer's needs. However, they may not be easily tuned to improve *both* the recall and precision rates *at the same time*. (See Section 1.4.3 on why a *filtering* approach normally cannot improve precision and recall simultaneously.) And, in general, they have a few other problems to be resolved.

1.4.2 Problems with Segmentation-Merging Using Known Words Only

First of all, some of the unknown words cannot be recovered by using the 'merge' operation. In fact, it is known in [Chiang 92, Lin 93a, Lin 93b] that, when there are unknown words in the text corpus, bad segments are generated not only due to over-segmenting unknown words into shorter segments but also due to *incorrectly merging* known words and/or single characters into longer segments [Lin 93b]. For instance, '<u>土地</u> <u>公有</u> <u>政策</u>' may be segmented incorrectly into '<u>土地公</u> <u>有</u> <u>政策</u>', in which '土地公' is an over-merged string, if '公有' is an unknown word and '土地公' is already registered in the system dictionary. Although it is possible for the 'merge' operation to recover unknown words which suffer from the over-segmentation error for further justification, the unknown words (such as '公有' in the last example) which suffer from the over-merging error, however, cannot be recovered by simply using the ‘merge’ operation.

Second, the filter cannot benefit from the contextual constraints over *unknown words* to produce better segmentation (and unknown word candidates), because suspected unknown words do not participate in the segmentation process. Under such circumstances, a large number of unknown word candidates (including many spurious word candidates) will be generated by the merge operation and then submitted to the filter. Many such spurious words, however, cannot be rejected by the filter since it is the contextual constraints, not the association features used by the filter, that reject such spurious words. With our corpus, for instance, such randomly merged unknown word candidates amount to more than 2 million strings (of 2-4 characters). The segmentation-and-merging process (using only known words for segmentation) may thus result in low system precision.

For example, the segmentation pattern ‘(獲) 省都委會 同意’ (‘gain the approval of the Provincial City Development Committee’) is identified by our lexicographer as the preferred segmentation. It will be segmented correctly if the (abbreviational) unknown word ‘省都委會’ is included in the system dictionary and thus participates in the segmentation process. In this case, no spurious word candidates will be submitted to the filter. On the other hand, if we only use known words for segmentation, the above string will be segmented into ‘(獲) 省 都 委 會同 意’. And by merging the short segments, we will have the spurious word candidates ‘省都’ (‘capital of the Province’), ‘省都委’ (‘committee members of the Provincial City Development Committee’), ‘都委’ (‘committee members of the City Development Committee’), ‘都 委會同’ (‘the committee members of the City Development Committee call a meeting to...’), ‘委會同’, ‘委會同意’, and ‘會同意’ (‘... will approve’); many such spurious word candidates will be accepted by the filter as legal words (since they might be highly associated). However, they should actually not be extracted in the current context if we agree that ‘省都委會’ and ‘同意’ are the only and the most preferred lexical entries in the above phrase.

The segmentation-and-merging scheme will thus degrade the precision performance of the system due to the introduction of a large number of spurious words. Such spurious words, however, will not be generated if highly likely potential unknown word candidates, such as '省都委會', are added to the system dictionary and participate in the segmentation process. In this case, only those unknown word candidates that are preferred by the segmentation process will survive and be submitted to the filter for further justification.

Third, a separated ambiguity resolution step might be required, due to the merging step, to remove overlapping ambiguities such as the previous '漁業區' and '區附近' example. Normally, the disambiguation step only compares adjacent candidates in a local context to decide which candidate to survive, without using the contextual constraints over the whole sentence to see whether the resultant segmentation is the desired segmentation pattern. In other words, such a disambiguation process does not choose the candidates to maximize the overall likelihood of the corpus. Therefore, this extra disambiguation process may incorrectly disambiguate some overlapping ambiguities which need be resolved using the contextual information, and thus may degrade the system performance.

For instance, '彰化 縣警 刑警隊 少年組' may be segmented into '彰化 縣 警 刑警隊 少年組' due to the lack of the unregistered new word '縣警' ('county police office'). After merging the segments, the two overlapping candidates '彰化縣' and '縣警' may both be qualified by the filter as words, but the ambiguity resolution step may reject '縣警' as a new word since it is a weaker competitor than '彰化縣' in the corpus. However, although '縣警' might be a weaker competitor than '彰化縣', it is very unlikely that: '彰化縣 警 刑警隊 少年組' is a good segmentation pattern, because the single character '警' is rarely used as a word by itself. In other words, if we know that the joint likelihood for '彰化縣' and '警' (in terms of the product of the their individual probabilities $P(\text{彰化縣}) \cdot P(\text{警})$) is much smaller than the joint likelihood of '彰化' and '縣警' (with likelihood $P(\text{彰化}) \cdot P(\text{縣警})$), we should not discard the new word '縣警' even though it is a weaker competitor than its neighbor. This means that such a (merging-)filtering-and-disambiguation process may prefer strong competitors regardless of the context. If, on the other hand, the potential unknown word candidates '彰化縣' and '縣警' are both included in the system dictionary, the additional disambiguation process would be unnecessary. The above problem can then be avoided.

In summary, the above problems could be overcome if an *augmented* dictionary, which contains *potential unknown words* (in addition to known words), can be used during the segmentation process to get a better segmentation. Under such circumstances, the improved segmentation result will contain highly likely unknown word candidates. Accordingly, the precision of the filter will be benefited from the better segmentation generated by using such an enlarged dictionary. For this purpose, an *augmented dictionary,* which includes both *potential unknown words* in the input corpus and known words in the system dictionary, will be used in our system in the segmentation process; and the probabilities associated with such potential unknown words will be jointly trained with other known words to globally maximizes the likelihood of the input text. Thus, the separated merging and disambiguation processes in conventional approaches will no more be necessary.

1.4.3 Problems with the Non-Iterative Scheme

Although the inclusion of potential unknown words in the segmentation process resolves the above mentioned problems of the segmentatiom-merging-filtering-disambiguation scheme, segmentation and filtering are still two independent steps, and thus they cannot enhance the performance of each other simply by cascading them.

To address this issue, first note that the performance of the segmentation module greatly depends on whether the augmented dictionary is close to the ideal dictionary (which contains *all and only* those words in the input corpus.) On the other hand, the performance of the filter depends greatly on whether its model is close to the true (lexicographer's) model and whether the model parameters are reliably estimated. We can therefore improve the system performance by improving the augmented dictionary and/or improving the parameters of the classifier.

Initially, it is obviously impossible to construct an augmented dictionary that contains all and only those words in the input corpus. Therefore, it is not surprising that some of the unknown word candidates identified by the segmentation module will still be spurious, although many of them might be the correct unknown words already. In addition, it is impossible, initially, to estimate the model parameters of the filter reliably since the filter usually makes its judgement based on its statistical model about the words and non-word n-grams, but we are not sure which n-grams are words and which are not, except for the words in the system dictionary. (For the Chinese unknown word extraction task, an n-gram refers to a group of n characters.)

However, we can use the filter to remove spurious words from the augmented dictionary and thus prevent them from appearing in the best segmentation pattern. This is possible since such spurious unknown word candidates, which are qualified by the word segmentation process simply because they co-occur frequently, may be rejected by the filter since they may not be strongly associated according to the criteria of the filter. (For instance, the filter may use a normalized co-occurrence frequency, instead of the co-occurrence frequency alone, to evaluate the strength of association). On the other hand, we can also use a well-segmented corpus to help estimating the parameters of the statistical model better (for instance, by moving those highly likely unknown words from the non-word class to the word class so that our statistical knowledge about the distributions of the word/non-word classes are better justified). Therefore, to improve the system performance further, we can form a feedback path to refine the augmented dictionary using the association features of the filter, and refine the model parameters of the filter based on the segmentation results. Such a process could then be applied *iteratively* to enhance the individual performance of the two modules, and thus the preformance of the whole system.

For example, the phrase '(移送)台中少年法庭審理' ('(send to) the Taichung Youth-Court for investigation') will be segmented as '(移送) <u>台中少年</u> <u>法庭審理</u>' ('(send) <u>the young boy who lives in Taichung</u> <u>to the court for investigation</u>') initially due to the high frequencies of the potential unknown words '<u>台中少年</u>' and '<u>法庭審理</u>'. If we use the association metrics of the filter to remove these two spurious words from the augmented dictionary, the segmentation results will be progressively refined. In fact, by removing the first segment '台中少年' from the augmented dictionary using word association metrics, the best segmentation becomes '(移送) <u>台中</u> <u>少年</u> <u>法庭審理</u>'; such a refined segmentation pattern allows us to better re-estimate the model parameters of the filter, and thus results in further deletion of the string '法庭審理' from the augmented dictionary; the deletion of '法庭審理' finally gives us the correct segmentation '(移送) 台中 少年法庭 審理'. In this process, we not only improve the precision (due to the filtering operations) but also improves the recall (by applying additional segmentation sessions, iteratively.)

Therefore, an *iterative* scheme is proposed here to fully integrate the contextual information used by the segmentation module and the association features used by the filter. In this iteratively integrated scheme, the filter is used to rank potential unknown words such that very unlikely potential unknown words can be removed from the augmented dictionary, and the parameters of the filter is improved according to the better and better estimated word population statistics acquired from better segmentation results.

With such an iterative scheme, the segmentation output is expected to be better and better by using the progressively refined augmented dictionary, which is acquired by using the association information to filter out inappropriate candidates. By iteratively refining the augmented dictionary and thus the segmentation output, we can also expect to tune the filter model and its parameters better and better using the progressively refined segmentation output.

There is one important point that is worth mentioning here. It is possible simply to reject the spurious words (such as '台中少年' and '法庭審理') by the filter using other association features, and thus improve the *precision* rate, like conventional non-iterative filtering approaches do. However, simply rejecting such candidates won't tell us what they really should be, and thus won't be helpful for improving the *recall* rate. More precisely, successful filtering will only improve the precision performance, *not* the *recall* rate; however, unsuccessful filtering will degrade both precision and recall. None of such filtering operations will improve the *recall* rate. Hence, conventional non-iterative filtering approaches usually cannot improve the precision and recall simultaneously. In contrast, the iterative scheme proposed here provides a way to recall the real new words (converted from some truncated spurious words) in later segmentation processes. In this way, the *precision* will be improved by *filtering*, and *recall* will be improved by *re-segmentation*. We can then expect to improve *both* precision and recall in the iterative scheme, without sacrificing precision for recall or *vice versa*. Such advantages are unlikely to be fully utilized in a non-iterative scheme.

To sum up, the general non-iterative segmentation-merging-filtering-disambiguation scheme is incapable of recovering the over-merging type of segmentation errors; the merging operation may also introduce many randomly merged segments to the filter; and a separated disambiguation process might be needed for resolving overlapping ambiguities. With the extra disambiguation process, the system performance might be degraded for some ambiguous pairs which need contextual information to resolve. Furthermore, due to the non-iterative nature of this scheme, the segmentation module and the filter module cannot help each other to acquire better performance; most likely, the precision and recall will not be improved simultaneously if the two modules are simply cascaded, since no feedback path is provided to recall the real unknown words corresponding to those rejected candidates.

To resolve such problems, an augmented dictionary, which contains potential unknown words, is used in the segmentation process. The augmented dictionary of the segmenter is refined by the filter. The model parameters of the filter are also progressively refined by using the word and non-word knowledge acquired from the progressively refined word segmentation output. Such progressive refinement is conducted through an iterative scheme to re-segment the input corpus and re-estimate the filter parameters. In this iterative process, the precision is improved by truncating inappropriate candidates in the augmented dictionary, and the segmentation process provides a way to recall the real new words for such truncated candidates; *both* precision and recall are thus improved progressively.

1.4.4 The Filter Design Problems

The filter for the Chinese unknown word extraction task suffers from essentially the same problems in filter (or classifier) design as in the English compound extraction task. Furthermore, we don't really have a pre-annotated corpus for training the classifier parameters under the current unsupervised mode of training; we therefore can't expect to achieve the best performance by using an absolute threshold for filtering, and we can't adjust the classifier parameters by applying adaptive learning.

To overcome these problems, a Likelihood Ratio Ranking Module (LRRM), based on the same two-class minimum error classification model for the English compound word extraction task (Section 2.4), is used as our ‘filter’ to combine all available association metrics into one *joint* association measure, namely the log-likelihood ratio (LLR), instead of using different filters and heuristic thresholds to independently apply the word association metrics one-by-one; moreover, such a measure is used for ranking the candidates in terms of the degree of association, so that we can tell which candidates are more likely (or more unlikely) to be words with respect to other candidates (instead of asserting which candidates are qualified words by checking their association metrics against some thresholds).

With such a relative ranking index, only a small fraction (say 5%) of the most unlikely unknown word candidates are truncated from the augmented dictionary, and only a small fraction of the most likely unknown word candidates are used for updating the filter parameters. In other words, the filter is not using an absolute threshold value for filtering, instead, a *relative* mode for filtering is adopted. Therefore, the proposed method will not heavily depend on the determination of an ‘optimal’ threshold (in whatever sense) to improve *both* the precision and recall rates, in contrast to other systems which heavily depend on the thresholds for tuning the precision or recall. This strategy is particularly useful for the current task, since we are operating in the unsupervised mode of operation, in which the classifier parameters may not be reliably estimated.

Because the ‘classifier’ now serves simply as a ranking device for supplementing the segmentation module, the unknown word extraction task is modeled mainly as a segmentation task. In our model, an *augmented dictionary*, which contains known words in the system dictionary and *suspected unknown words* in the input corpus, is formed for segmentation. The extraction task tries to extract the embedded words in the input corpus by maximizing the likelihood of the input corpus in terms of the words and potential unknown words in the augmented dictionary. This is accomplished with a Viterbi word segmentation module (Sections 4.3 and 4.4), which segments the input corpus into various segmentation patterns by looking up the augmented dictionary, and selects the pattern with the highest likelihood probability. And the ranking module serves to improve the augmented dictionary in an iterative manner. (In the following chapters, we will sometimes use the terms 'filter', 'likelihood ratio ranking module' and 'two-class classifier' interchangeably for convenience.)

1.5 Main Contributions

Briefly, two major lexicon extraction techniques, which are capable of improving the joint precision-recall performance systematically, are developed in this dissertation. (1) For the English compound word extraction task, a two stage optimization scheme is developed; the first optimization stage aims at minimizing the classification error rate, and the second optimization stage focuses on the maximization of user specified criterion functions of precision and recall, starting from the minimum error status. In particular, a non-linear adaptive learning method, which adjusts the statistical parameters of the minimum error classifier, is proposed in this task. The main philosophy in this kind of optimization technique is to achieve the best performance by adjusting the classifier parameters. (2) For the Chinese unknown word identification task, we not only try to improve the precision rate by improving the classifier parameters (and the parameters for the segmentation module) but also try to recover the correct word if the the segmentation module incorrectly identified a word in a previous iteration; an iterative method is thus developed for these purposes. In this manner, the precision is improved by using the classifier (or filter) to filter out inappropriate candidates in the segmentation output, and the recall is improved by recovering those real words corresponding to such inappropriate candidates using the global contextual constraints. Both precision and recall can thus be seen to increase monotonically without sacrificing precision for recall or *vice versa*. The main philosophy that deserves appreciation in such an optimization technique is to *recover errors* that were made in previous iterations by using other information, in addition to improving the classifier itself (and other system modules) by adjusting the parameters.

1.6 Organization of this Dissertation

The dissertation is organized as follows. Chapters 2-3 study the optimization techniques in the two stage optimization scheme for English compound word extraction. Chapters 4-5, on the other hand, study the effects of integrating global contextual constraints and association features iteratively for recovering segmentation errors made in previous iterations.

In Chapter 2, we first define the optimization criteria for the lexicon extraction tasks. The existence problem of the optimal classifier for such optimization criteria, and a two-stage scheme for maximizing the classifier performance then follow. The minimum error classifier, which is to be used in the first stage of the two-stage optimization procedure is then addressed. The features used in the current work are discussed afterward, including part-of-speech related features which encode syntactic information quantitatively. The experiment environments for the compound word extraction task is described, and the performance of the baseline system using only one feature is demonstrated. The issues for achieving minimum error systematically are then discussed in sequence, including joint integration of multiple features to minimize classification errors by feature selection and by considering feature correlation, instead of using multiple features with a step-by-step filtering approach. Chapter 3 outlines the precision and recall optimization technique which maximizes the system performance by adjusting the parameters of the classifier. A nonlinear learning method, which operates in batch mode of learning, is developed in this chapter. Learning results are given to demonstrate the effects of the proposed optimization techniques, including the effects to improve different optimization functions specified by different users. A section is also devoted to some subtle points on learning when combining continuous-valued word association features with discrete-valued features. Summary is then given.

Starting from Chapter 4, our focus will be moved to the more complicated Chinese unknown word extraction task. The suggested system architecture for integrating contextual information used by the segmentation module and association metrics used by the classifier (or ranking module) is outlined first. The word segmentation model which is used to apply global contextual constraints over the new word candidates is then discussed. A Viterbi training method is proposed to maximize the likelihood of the segmentation patterns; preliminary results for such a baseline system is evaluated. Afterward, the way to use the classifier for truncating inappropriate new word candidates from the augmented dictionary, and a cascaded scheme for combining the classifier with the segmentation module is surveyed. In Chapter 5, an iterative scheme for jointly improving the precision and recall performance is proposed to overcome problems with the non-iterative scheme. The effects of such an iterative improvement approach are evaluated. Quantitative analyses for the acquired words and the remaining errors are then given. Finally, summary is given, followed by future research issues.

In the following chapters, we will survey the English compound extraction task and Chinese unknown word identification task in sequence, and explore the precision-recall maximization techniques for these two languages using the above outlined optimization techniques.

Chapter 2
English Compound Word Extraction

A compound word extraction method, which is capable of optimizing the joint precision and recall performance automatically, is proposed in the following chapters for extracting English compound words within a text corpus. Traditional terminology extraction methods rely heavily on simple filtering-and- thresholding methods, which are unable to optimize a pre-specified criterion function of the precision and recall performance. A two-stage optimization scheme is thus developed in the following chapters to optimize the joint precision-recall performance. In particular, a nonlinear learning method, which adjust the system parameters of a (minimum error) classifier in batch mode, is developed for this purpose.

The extraction problem is modeled as a two-class classification problem, in which word n-grams are classified as compound or non-compound by using word association metrics as well as parts of speech information. The extraction task is conducted in two optimization stages. The first stage aims at minimizing the classification errors of the classifier, and the second stage focuses on maximizing the joint precision and recall performance. The first optimization step is resolved by using a likelihood ratio test approach, which is known to be the most powerful test in statistics as well as the optimal decision rule for minimum error classification. For the second stage, a non-linear learning algorithm is developed for optimizing the precision and recall rates. Optimization on *weighted precision & recall* (WPR) and *F-measure* (FM), in particular, are surveyed in detail.

2.1 Optimization Criteria for Lexicon Extraction

Before addressing the optimization techniques, the optimization criteria for the current task are defined first. In a compound extraction task, it is desirable to extract from the corpus as many real candidates as possible; in addition, the extracted compound word list should contains as little 'false alarm' (i.e., incorrect candidates) as possible. The ability to extract real candidates in the corpus is defined in terms of the recall rate, which is the percentage of real compounds that are extracted to the compound list by the classifier; on the other hand, the ability to exclude false alarm from the extracted compound list is defined in terms of the precision rate, which is the percentage of real compounds in the extracted compound list. Let $n_{\alpha\beta}$ be the number of class-α input tokens which are classified as class-β (α, β = 1 for compound, and 2 for non-compound, respectively), and, let n_1 represent the number of real compounds in the corpus. The precision *p* and recall *r* are defined as follows:

$$p = \frac{n_{11}}{n_{11} + n_{21}}$$

$$r = \frac{n_{11}}{n_{11} + n_{12}} = \frac{n_{11}}{n_1}$$

The precision and recall rates are, in many cases, two contradictory performance indices especially for simple filtering approaches. When one of the performance index is raised, another index might degrade. To make fair comparison, a joint performance indice or criterion function O(p,r) of the precision (p) and recall (r) rates is usually used to evaluate the system performance, instead of evaluating precision or recall alone.

In the following sections, the *weighted precision & recall* (WPR) and the *F-measure* (FM), defined as follows, will be adopted as the optimization criteria. The proposed optimization mechanism, however, can be applied to other functions of the precision and recall rates which characterize the user preference toward a particular precision-*vs*-recall profile. The weighted precision and recall (WPR), which reflects the average of these two indices, is proposed here as the weighting sum of the precision and recall rates:

$$WPR\left(w_p : w_r\right) = w_p * p + w_r * r \qquad (w_p + w_r = 1)$$

where w_p, w_r are the weighting factors for precision and recall, respectively. The F-measure (FM) [Appelt 93, Hirschman 95, Hobbs 96], defined as follows, is another joint performance metric which allows lexicographers to weight precision and recall differently:

$$FM(\beta) = \frac{\left(\beta^2 + 1\right) p r}{\beta^2 p + r}$$

where β encodes user preference on precision or recall. When β is close to 0 (i.e., FM is close to p), the lexicographer prefers the system with higher precision; when β is large, the lexicographer prefers the system with higher recall. We will use Wp=Wr=0.5 and β=1, throughout this work, which means that no particular preference over precision or recall is imposed. If β=1, FM reduces to $\frac{2pr}{p+r}$, which appreciates the balance between precision and recall in the sense that equal precision and recall is most preferred if $p+r$ is identical. Our goal is to design an optimal classifier which could maximize the WPR and FM.

2.2 Task Definition for Optimal Classifier Design

The extraction problem can be regarded as a two-class classification problem in which each n-gram candidate is assigned either the compound label or the non-compound label based on the feature vector **x** associated with the candidate. To design a compound extractor is therefore equivalent to designing a discrimination function $g(\mathbf{x};\Lambda)$ (which is capable of scoring how likely a candidate comes from the compound class), and using a set of decision rules (in terms of $g(\mathbf{x};\Lambda)$) to decide whether an n-gram candidate is a compound. (The symbol Λ in the discrimination function refers to the parameters of the discrimination function, such as the means or variances of the probability density functions of a statistical model.)

Different discrimination functions and decision rules will classify the input candidates differently, and thus have different performance in terms of a performance criterion. Designing an optimal classifier for a particular criterion function is therefore equivalent to finding a partition of the feature space into the decision regions for the compound class and non-compound class; feature vectors belonging to the compound decision regions are classified as compound, otherwise, they are classified as non-compound. Our main task is therefore to design an optimal classifier (or equivalently the corresponding discrimination function $g^*_{O(p,r)}(\mathbf{x};\Lambda)$) which could maximize an objective criterion function O(p,r) of the precision (p) and recall (r) rates.

2.3 A Two-Stage Strategy for Precision and Recall Optimization

Given the underlying distributions, $f(\mathbf{x}|\mathbf{C})$ and $f(\mathbf{x}|\overline{\mathbf{C}})$, of the feature vectors **x** in the compound class (**C**) and non-compound class ($\bar{\mathbf{C}}$), it is possible to estimate the error probabilities associated with any decision region (or equivalently, any threshold, decision rules or statistical tests which could be used to define such a region) for a class. Therefore, it is possible to derive the optimal classifier from the two class distributions for some simple criteria, such as minimum classification error. In fact, optimal classifiers for minimum classification error had been well studied in the speech, communication and pattern recognition communities [Devijver 82, Juang 92], and the decision rule for finding the optimal decision regions that minimizes the expected probability of classification error turns out to be a likelihood ratio test in the 2-class classification case [Devijver 82].

However, since WPR and FM are non-linear functions of classification errors (i.e., a non-linear function of n_{12} and n_{21}), it is generally hard to find a simple analytical discrimination function $g^*_{O(p,r)}(\mathbf{x};\Lambda)$ for testing whether an n-gram is a compound or not, such that the joint performance O(p,r) is maximum. Therefore, a two stage optimization scheme is proposed here for searching a local optimum in terms of the criterion function.

Because small error rate normally implies good precision and recall performance (as will be explained quantitatively shortly), it is desirable that the optimization process starts from a point in the parameter space which could achieve the minimum error rate performance. The compound extraction task is thus attacked here in two stages to optimize a user specified criterion function of precision and recall. In the first stage, a minimum error classifier, $g^*_e(\mathbf{x};\Lambda)$, whose optimization criterion is closely related to the precision-recall performance, is used as the baseline classifier. In the second stage, a learning method is applied, starting from the minimum error status, to optimize a user-specified criterion function of the recall and precision rates by adjusting the parameters of the classifier according to mis-classified instances.

The first step serves one main purpose: it determines the appropriate thresholds (or, more precisely, the decision boundaries) in the feature space so that as little misclassification is attained as possible. The second stage, on the other hand, adjusts the parameters of the classifier to achieve a local optimum of the joint precision-recall performance, starting from the minimum error status, instead of optimizing the precision and recall from arbitrary decision boundary.

In other words, we are not trying to find some simple analytical discrimination function which are capable of identifying the optimal decision boundaries for precision-recall optimization. Instead, we first establish reasonably optimized decision boundaries by using the simple discrimination function for the minimum error classifier, and then modify the decision boundaries by changing the parameters of the distribution functions of the minimum error classifier to maximize the joint precision-recall performance.

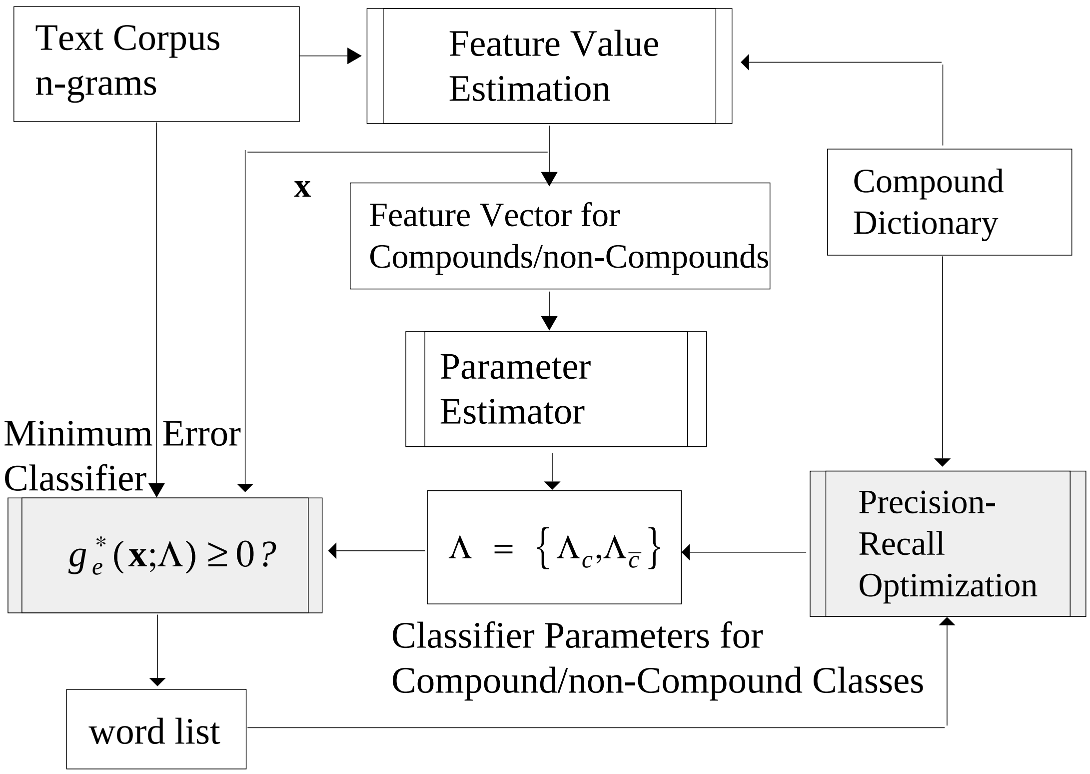

Figure 1 Supervised training of classifier parameters for precision-recall maximization.

Figure 1 shows the block diagram for training the parameters of such a (sub)optimal classifier. In the training flow, the n-grams in the training text corpus are manually inspected, and those real compounds within the text corpus are collected to construct a compound dictionary. The feature vectors associated with the n-grams are divided into the compound and non-compound classes according to the compound dictionary. The parameters for the compound class (Λ_c) and non-compound class ($\Lambda_{\bar{c}}$) are estimated from the distributions of the two classes. The training n-grams are then classified by the minimum error classifier. The result is compared with the compound dictionary afterward. Those misclassified n-grams are then used to adjust the parameters iteratively so that the criterion function is maximized. (The two shaded blocks correspond to the two optimization stages by which the classifier parameters are trained.)

The minimum error classifier is adopted at the first optimization stage since it's criterion, namely minimum classification error, is closely related to the precision and recall performance; reducing classification error, in general, will improve precision or recall. In addition, it is relatively easy to implement a minimum error classifier [Devijver 82, Juang 92], and we don't want to optimize the precision-recall starting from an arbitrary state. It is reasonable to expect that a better local maximum, in terms of precision and recall, is highly likely to be found near the parameter point that archives the minimum error classification.

To see the relationship between the error rate and the precision/recall rates quantitatively, first note that $p = \left(1 + n_{21}/n_{11}\right)^{-1}$ and $r = \left(1 + n_{12}/n_{11}\right)^{-1}$. The precision and recall can thus be improved by reducing n_{21} and n_{12}, respectively. (Note that reducing n_{12} is equivalent to increasing n_{11} because their sum is the constant n_1.) Since, the error rate is propotional to $n_{12} + n_{21}$, the minimum error rate (i.e., minimum $n_{12} + n_{21}$) status is a good initial point for further optimizing the precision, recall or WPR performance. As far as F-measure is concerned, maximizing the F-measure is equivalent to minimizing $\left(n_{12} + n_{21}\right)/n_{11}$. (See Appendix I for the proof.) Therefore, it is also appropriate for using minimum error as the criterion of the first optimization stage. In fact, if we plot the WPR (or FM) graph as a function of n_{21} and n_{12}, moving toward minimum error tends to have higher WPR (or FM) in general.

In spite of such close relationship between error rate and precision-recall, the minimum error criterion, eventually, is not equivalent to the maximum precision-recall criterion for a given function of precision and recall. Therefore, a learning method is required to further maximize the criterion function of precision and recall. This could be implemented by changing the parameters of the classifier according to the mis-classified instances so that mis-classified instances are likely to be correctly identified in the next iteration. Unlike other learning algorithms for minimizing classification error, which adjust the parameters once a classification error is detected, the learning method for precision-recall optimization cannot adjust the classifier parameters by each mis-classification instance, because the precision and recall (and WPR, FM) are non-linear function of the error counts n_{21} and n_{12}.

There are several issues related to the design of the minimum error classifier and the learning algorithm. As mentioned previously, it is desirable to use features that encode syntactic information, such as parts of speech in the feature set to reject highly associated non-compound candidates; it is also desirable to use multiple features jointly to enjoy all the information contained in the features. However, the feature correlation among the features must be carefully handled in order to model the distributions of the features properly, and redundant or non-discriminative features should be removed when combining the features. Furthermore, the parameters must be estimated in a way as to minimize the classification errors. As far as the learning algorithm is concerned, a learning rule that ensure a non-decreasing trend of the joint precision-recall performance is required. These issues will be addressed in the following sections.

2.4 A Two-Class Classification Model for Minimum Error Classification

A likelihood ratio test method, which was proved to be the most powerful test [Papoulis 90], can be used as the baseline classifier to achieve minimum classification error if the distributions of the feature vectors of the two classes are known. In fact, it implicitly implies the use of the optimal thresholds (or decision boundaries) which minimize the misclassification costs in Bayesian decision points of view [Devijver 82, Juang 92] if the cost for each misclassification is unity. In other words, it minimizes the probability of errors for two classes of known distributions.

To identify whether an n-gram is a compound or a non-compound, each n-gram is associated with a feature vector $\mathbf{x}$, it is then judged to see whether it is more likely to be generated from the compound class $\mathbf{C}$ or the non-compound class $\overline{\mathbf{C}}$ based on the following (log-)likelihood ratio:

$$\lambda = \frac{f(\mathbf{x}|\mathbf{C})P(\mathbf{C})}{f(\mathbf{x}|\overline{\mathbf{C}})P(\overline{\mathbf{C}})}$$

$$g(\mathbf{x}) \triangleq \log \lambda = \log f(\mathbf{x}|\mathbf{C}) - \log f\left(\mathbf{x}|\overline{\mathbf{C}}\right) + \log P(\mathbf{C}) - \log P\left(\overline{\mathbf{C}}\right),$$

where $P(\mathbf{C})$ and $P\left(\overline{\mathbf{C}}\right)$ are the prior probabilities of the two classes and $f(\mathbf{x}|\mathbf{C})$ (resp. $f\left(\mathbf{x}|\overline{\mathbf{C}}\right)$) is the probability density function of the feature vector $\mathbf{x}$ in the compound (resp. non-compound) class. If the likelihood ratio $\lambda \geq 1$ (i.e., the discrimination function $g(\mathbf{x})$, or the log-likelihood ratio, $\log \lambda \geq 0$) for an n-gram, then it is classified as a compound; otherwise, it is classified as a non-compound. The model parameters for the two classes are referred to as Λ_c and $\Lambda_{\bar{c}}$, respectively; they correspond to the means, variances, prior probabilities, etc. (depending on how the density functions are formulated), in the above formula.

2.5 Features for English Compound Word Extraction

The performance upper bound of the classifier depends on the distribution of the input feature vector $\mathbf{x}$. Many statistical features are used in various applications in the literatures. In particular, the *normalized frequency* (NF) [Wu 93, Su 94b] of an n-gram, the *mutual information* (MI) [Church 90, Su 91, Chang 95] among the words within an n-gram, the *dice* metric (D) [Smadja 96] among the words of the n-gram, the *contextual entropy* (H) [Tung 94, Chang 95] of the neighboring words of the n-grams, are used in the current classification task. The definitions of such features will be given in the following sections.

In addition, part of speech (POS) is an important syntactic feature for extracting compounds. For instance, many compound words are associated with the part of speech patterns: [noun noun], [adjective noun] (for bigrams) or [noun noun noun], [adjective noun noun] (for trigrams). Previous frameworks [Wu 93, Su 94] show that simple word association metrics are useful for extracting highly associated compound words. However, many non-compound n-grams, like ‘is a’, which have high association and high frequency of occurrence are also recognized as compounds. Such n-grams could be rejected if syntactic information is available.

Therefore, we will propose a new feature, referred to as *part of speech discrimination* metric (Dpos), in this dissertation, and include the *part of speech* (POS) patterns associated with the word n-grams into the likelihood functions. When all such features are used, we will have the following discrimination function:

$$\begin{aligned} g(\mathbf{x}) &= \log\frac{f\left(NF,MI,D,H,D_{pos},POS|\mathbf{C}\right)P(\mathbf{C})}{f\left(NF,MI,D,H,D_{pos},POS|\overline{\mathbf{C}}\right)P\left(\overline{\mathbf{C}}\right)} \\ &\approx \log\frac{f\left(NF,MI,D,H,D_{pos}|\mathbf{C}\right)P(POS|\mathbf{C})P(\mathbf{C})}{f\left(NF,MI,D,H,D_{pos}|\overline{\mathbf{C}}\right)P\left(POS|\overline{\mathbf{C}}\right)P\left(\overline{\mathbf{C}}\right)} \end{aligned}$$

Since such features might contain redundant information, only a subset of the features will be automatically selected with a feature selection mechanism for classification.

The part of speech related features are proposed in this dissertation to encode syntactic information so that syntactic information could be integrated with other simple word association metrics in a simple and effective way, without resorting to complicated syntactic processing. For simplicity, we assume that the part of speech patterns are independent of other continuous-valued association features, and the likelihood probabilities P(POS|**C**) and P(POS|$\overline{\mathbf{C}}$) can be estimated independently of other features. The definitions of such features are introduced in the following sections.

2.5.1 Normalized Frequency (NF) [Wu 93]

The normalized frequency for the i^{th} n-gram is defined as:

$$r_i = \frac{f_i}{\bar{f}}, \qquad \bar{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$$

where f_i is the total number of occurrences of the i^{th} n-gram in the corpus, and $\bar{f}$ is the average number of occurrence of all the entries. In other words, r_i is the normalized frequency with respect to the average frequency $\bar{f}$. Intuitively, a word n-gram that is more frequently used than the average frequency is more likely to be a compound than a rarely used n-gram. Furthermore, it may not worth the cost of entering the compound into the dictionary if it occurs only a few times. The normalized frequency is therefore used as a feature for compound extraction.

2.5.2 Mutual Information (MI) [Church 90, Su 91]

Mutual information is a measure of word association. It is the ratio between the joint probability for a group of words to appear in the same n-gram window and the probabily for such words to occur in the same window independently. The bigram mutual information $I(x;y)$ is known as [Church 90]:

$$I(x;y) = \log_2 \frac{P(x,y)}{P(x) \times P(y)}$$

where x and y are two words in the corpus. The mutual information of a trigram is defined as [Su 91]:

$$I(x,y,z) = log\frac{P_D(x,y,z)}{P_I(x,y,z)} = log\frac{P(x,y,z)}{P_I(x,y,z)}$$

$$P_I = P(x)P(y)P(z) + P(x)P(y,z) + P(x,y)P(z)$$

where $P_D(x,y,z) \equiv P(x,y,z)$ is the joint probability for x, y, z to appear jointly as a group of words in a trigram window, and $P_I(x,y,z)$ is the probability for x, y, z to appear, independently, as a group by chance. Note that the three product terms in $P_I(x,y,z)$ correspond to three different ways in which the constituents of the trigram appear in the same trigram window by chance; $P_I(x,y,z)$ is the total probability of the various possible combinations. In general, $I(\cdot)$ >> 0 implies that the words in the n-gram are strongly associated. Otherwise, their appearance as one group of words may be simply by chance.

2.5.3 Dice Metric (D) [Smadja 96]

The dice metric is commonly used in information retrieval tasks [Salton 83] for identifying closely related binary relations. The dice metric for a pair of words x, y is defined as follows [Smadja 96]

$$D_2(x,y) = \frac{P(x=1,y=1)}{\frac{1}{2}[P(x=1)+P(y=1)]} ,$$

where x=1 and y=1 correspond to the events that x appears in the first place and y appears in the second place of a bigram respectively. The dice metric is another indication of word association which is similar to the mutual information metric. Intuitively, the dice metric is the likelihood ratio between the joint probability for two words (or events) to occur simultaneous and the average probability for each individual word (or event to) occur in bigram pairs. Therefore, a high dice value tends to mean that x and y are highly associated. It was suggested, in some researches (e.g., [Smadja 96]), that this feature is a better indication of word association than the (specific) mutual information metric in some cases since it discards less informative events corresponding to the x=0 and y=0 (0-0 match) cases in estimating the word association for a bigram.

We can also define the dice metric for triple relations following the same spirit in defining the trigram mutual information. However, note that in defining the bigram dice metric, the joint probability $P(x=1,y=1)$ is normalized with respect to the *average* of the marginal probabilities, $P(x=1)$ and $P(y=1)$, of the constituents instead of the *product* of the marginal probabilities (i.e., the probability of independent occurrence). Therefore, we have three different ways to normalize the joint probability with respect to the averages of the marginal constituent probabilities as follows:

$$\frac{P(x=1,\ yz=1)}{\frac{1}{2}[P(x=1)+P(yz=1)]},\ \frac{P(xy=1,\ z=1)}{\frac{1}{2}[P(xy=1)+P(z=1)]},\ or\ \frac{P(x=1,\ y=1,\ z=1)}{\frac{1}{3}[P(x=1)+P(y=1)+P(z=1)]}$$

where $P(x=1,y=1,z=1)$ is the probability that x, y and z appear simultaneously in the first, second, and third places of a trigram, $P(xy=1)$ (i.e., $P(x=1,y=1)$) is the probability that x and y appear simultaneously in the first and second places of a trigram, and $P(yz=1)$ (i.e., $P(y=1,z=1)$) stands for the probability that y and z appear in the second and third places of a trigram simultaneously. (Note that the first two normalized metrics are simply the bigram dice metrics for [x, yz] and [xy, z], respectively.) If any of the above three normalized association metrics is small, then the trigram is likely to belong to different words. Therefore, we shall use the *minimum* of the three normalized likelihood ratios to indicate the association of the trigram. The trigram dice metric is then defined as follows.

$$
\begin{aligned}
D_3(x,y,z) &= min\left[\frac{2P(x=1,y=1,z=1)}{P(x=1)+P(yz=1)}, \frac{2P(x=1,y=1,z=1)}{P(xy=1)+P(z=1)}, \frac{3P(x=1,y=1,z=1)}{P(x=1)+P(y=1)+P(z=1)}\right] \\
&\triangleq \frac{P(x=1,y=1,z=1)}{P_I'} \\
P_I' &= max\left[\frac{1}{2}[P(x=1)+P(yz=1)], \frac{1}{2}[P(xy=1)+P(z=1)], \frac{1}{3}[P(x=1)+P(y=1)+P(z=1)]\right]
\end{aligned}
$$

The three terms in the bracket of the *min* operator indicate three different ways in which the three words do not belong to the same lexical entry. The *min* operator means to choose the weakest evidence of association for comparison with a threshold. If the weakest evidence of association is greater than a threshold, then the trigram dice measure gives a strong indication that the three words belong to the same lexical entry. Given the above definition, only those trigrams which are significantly more likely to appear simultaneously than the most probable combination of the constituents are considered as compound candidates.

2.5.4 Contextual Entropy (H) [Tung 94]

Since the neighboring words of a compound word tend to be more freely distributed than the constituents of the compound, some previous studies use such a feature to identify potential new words [Tung 94]. The left and right contextual entropies are defined respectively as follows:

$$
\begin{aligned}
H_L(x) &= -\sum_{w_i} P_L(w_i;x) \log P_L(w_i;x) \\
H_R(x) &= -\sum_{w_i} P_R(x;w_i) \log P_R(x;w_i)
\end{aligned}
$$

where w_i is the left or right neighboring word of an n-gram x, and the probability that w_i appear as the left or right neighbor of an n-gram x is represented as $P_L(w_i;x)$ and $P_R(x;w_i)$, respectively. If the contextual entropy is large, which means that the neighbors of x are randomly distributed, then x tends to be a lexical unit by itself; otherwise, x and w_i are likely to appear simultaneously, which implies that x is unlikely to be a lexical unit by itself. In the current work, we use the average of H_L and H_R as a single feature instead of using the two entropy metrics.

2.6 POS Discrimination (Dpos)

In addition to the above commonly used association features, parts of speech related features are desirable for encoding syntactic information. Two parts of speech related features are proposed in the following sections.

One way to use the POS information is to measure how similarly the candidate and the compound class are tagged with different POS patterns. Since an n-gram may be associated with different part of speech patterns under different context (some of which may have been incorrectly tagged in the tagging process), and each distinct POS pattern may not have the same frequency of occurrence, the distribution of such POS patterns provide us a way to measure their similarity. For instance, if the compound words are tagged as [noun noun] in 80% of the cases and as [adjective noun] in 20% of the cases, then a candidate which was tagged as [noun noun] (or [adjective noun]) in most of the cases is very likely to be a compound. In this dissertation, we thus suggest the following *POS Discrimination* metric for measuring the similarity or distance between a compound word candidate x_i and the compound word class, in terms of their tagged POS patterns. The *discrimination* metric [Blahut 87] is defined as follows in terms of the distribution, P_{ij}, of the POS patterns of the candidate and the distribution, P_j, of the POS patterns of the compound class:

$$D_{pos}(x_i;\{P_{ij}\},\{P_j\}) = \sum_j P_{ij} \log \frac{P_{ij}}{P_j}$$
$$P_{ij} \equiv P(j|w_i), \qquad P_j \equiv P(j)$$

where P_{ij} is the probability for the *i*th compound word candidate (or n-gram) to be tagged with the part of speech pattern j (such as a [noun noun] tag pair) and P_j is the probability for any compound word to be tagged as j.

Intuitively, the log-likelihood ratio of P_{ij} over P_j indicates how close or similar (in terms of probability of occurrence) the particular POS pattern j is, in comparison with the probability for the whole class. If the two probabilities are nearly identical, that is, $P_{ij} \approx P_j$, the log-likelihood ratio will be close to zero. Otherwise, the 'distance' will be large. The probability P_{ij} preceding the log-likelihood ratio is a weighting factor indicating how often such a 'distance' is observed; the discrimination metric is thus the expected distance between the two probability distributions of POS tagging patterns. When a compound word candidate has exactly the same distribution as the distribution for the compound class ($P_{ij} = P_j$ for all j), the 'distance' will be exactly zero. Therefore, we can gather the POS distributions of the n-grams, and use the distributions of such a distance measure in the two classes to see whether the candidate comes from the compound class.

Since this metric assumes continuous values, the distribution of this metric can be expressed in a parametric form and the parameters of the probability density functions can be estimated from a training corpus. We can thus easily incorporate such POS information for identifying compound terminologies with a few such parameters in a very simple and effective way.

2.7 Part of Speech Likelihood (Lpos)

Another way to include POS constraints quantitatively is to consider the joint likelihood of the POS patterns associated with an input n-gram in the two classes, i.e., P(POS|**C**) and P(POS|$\overline{\mathbf{C}}$). Intuitively, if a particular part of speech pattern associated with the n-grams, such as [noun noun], occurs more frequently in the compound class than it does in the non-compound class, then an n-gram which has high likelihood of being tagged as [noun noun] is likely to be a compound. Therefore, the likelihood ratio for an n-gram to be tagged with a particular part of speech pattern in the two classes is a useful indicator for classifying the n-gram.

In more general cases, the *joint* likelihood of the various POS patterns associated with an n-grams observed in the input text should be used for making better judgement, since an n-gram may be tagged with different part of speech patterns under different context, and each distinct POS pattern may not have the same frequency of occurrence. In addition, it is desirable to take the parts of speech of the neighboring words into consideration. Therefore, we will estimate the joint likelihood of the POS patterns associated with an n-gram candidate as follows.

Firstly, we define the *extended* POS pattern associated with an n-gram as the parts of speech tagged to the n-gram and the parts of speech of its neighboring words. For example, if a "noun-noun" bigram is tagged as [n n], and its left neighbor and right neighbor are tagged as ‘a’ (adjective) and ‘n’ (noun), respectively, then the compound candidate will be associated with an *extended* POS pattern, $\overline{L}_j$ = [a (n n) n]. (The tags enclosed by the parentheses are the tags for the bigram, the other tags to the left/right of the parentheses are the tags for the left/right neighbors. The over-head bar of the symbol is simply used to indicate an extended POS pattern, in which the left/right context are included.) Then the joint likelihood of the parts of speech patterns associated with an n-gram (with context taken into account), in a given class $\mathbf{C}$, can be estimated as follows, assuming that each POS pattern is tagged independently:

$$
\begin{aligned}
P(POS|\mathbf{C}) &= P\left(\{\overline{L}_j\}_{j=1,N}|\mathbf{C}\right) \\
&\approx \prod_{j=1}^{N} P\left(\overline{L}_j|\mathbf{C}\right) \\
&= \prod_{i=1}^{K} P\left(\overline{L}_i|\mathbf{C}\right)^{N\left(\overline{L}_i\right)} \qquad \left(\sum_{i=1}^{K} N\left(\overline{L}_i\right) = N\right)
\end{aligned}
$$

In the above expressions, N is the total number of occurrence of the n-gram in the text, K is the number of distinct (extended) POS patterns for the n-gram, and $N\left(\overline{L}_i\right)$ is the number of occurrence of a particular POS pattern $\overline{L}_i$ among the N occurrences. Note that, the log-likelihood ratio, for POS alone, is

$$
\begin{aligned}
\log\lambda &= \log\frac{P(POS|\mathbf{C})}{P\left(POS|\overline{\mathbf{C}}\right)} - \log\frac{P(\mathbf{C})}{P\left(\overline{\mathbf{C}}\right)} \\
&= \sum_{i=1}^{K} N\left(\overline{L}_i\right)\log\frac{P\left(\overline{L}_i|\mathbf{C}\right)}{P\left(\overline{L}_i|\overline{\mathbf{C}}\right)} - \log\frac{P(\mathbf{C})}{P\left(\overline{\mathbf{C}}\right)} \qquad \left(\sum_{i=1}^{K} N\left(\overline{L}_i\right) = N\right)
\end{aligned}
$$

which could be interpreted as the expected (i.e., average) likelihood ratio of the individual (extended) POS pattern

$$\log\frac{P(\overline{L}_i|\mathbf{C})}{P(\overline{L}_i|\overline{\mathbf{C}})}$$

If bigrams are more frequently tagged as [n n] following an 'a(djective)' and preceding a 'n(oun)', i.e., $\overline{L}_i$ = [a (n n) n], in the compound class than they do in the non-compound class, then the individual likelihood ratio for $\overline{L}_i$ will be much greater than 0; if otherwise, it will be much smaller than 0. In addition, if such a POS pattern occurs more frequently than other patterns (such as [a (a n) n]) in the corpus, then the corresponding log-likelihood ratio will be given a heavier weight ($N(\overline{L}_i)$). In fact, the factor

$$\sum_{i=1}^{K}\frac{N(\overline{L}_i)}{N}\log\frac{P(\overline{L}_i|\mathbf{C})}{P(\overline{L}_i|\overline{\mathbf{C}})} \qquad \left(\sum_{i=1}^{K}N(\overline{L}_i)=N\right)$$

could be interpreted as the information-theoretic metric known as the *discrimination* between two distributions [Blahut 87], which is commonly used as a metric of 'distance' between two probability distributions. In the current task, such a metric is used to formulate the distance between the two hypotheses in the two class classification problem. With such a formulation, it is easy to integrate POS information with other word association metrics, and optimize the extraction task by using such features.

To reduce the number of parameters, the probability for individual POS pattern, $P(\overline{L}_i|\mathbf{C})$, is approximated as the product of the transition probabilities between the adjacent tags in $\overline{L}_i$, since the tag of the current word is assumed to be highly correlated to the tag of its neighbor. In other words, for bigrams,

$$\begin{aligned} P(\overline{L}_i|\mathbf{C}) &= P([C_0\ (C_1\ C_2)\ C_3]|\mathbf{C}) \\ &\approx P_1(C_1)P_{10}(C_0|C_1)P_{12}(C_2|C_1)P_{23}(C_3|C_2) \end{aligned}$$

where C_1, C_2 refers to the POS tags for the first and second words in the bigram, C_0 refers to the POS of the left context, and C_3 refers to the POS of the right context, and $[C_0\ (C_1\ C_2)\ C_3]$ refers to the ith extended POS pattern for the bigram. In the above equations, $P_{ij}(C_j|C_i)$ refers to the conditional probability for the *j*th part of speech in the extended POS to be C_j given that the *i*th part of speech is C_i (i, j=0- 3 for bigrams), and $P_1(C_1)$ is the probability for the first part of speech of the n-gram to be C_1 .

For example, if a bigram is associated with an extended POS pattern of [a (n n) v], then the likelihood associated with such an extended POS pattern in the compound class would be simplified as:

$$\begin{aligned} & P([a\ \ (n\ \ n)\ \ v]|\mathbf{C}) \\ & \approx\ P_1(n)P_{10}(a|n)P_{12}(n|n)P_{23}\left(v|n\right) \end{aligned}$$

The formulation for trigrams can be simplified in a similar way.

Since this POS feature is discrete-valued, unlike other association features, which are continuous-valued, we will defer the techniques for combining these two different types of features in a later section after the techniques for combining those continuous-valued association features are surveyed.

2.8 Experiment Environments for Compound Word Extraction

To investigate the various models, a corpus of 23,016 sentences (188,267 words) is prepared. The corpus is collected from a technical manual for cars. It is first processed by a morphological analyzer to normalize every word into its stem form, instead of its surface form, to reduce the number of possible variants. Since parts of speech are used as a compound extraction feature, the text is tagged by a probabilistic part-of-speech tagger [Lin 92, Lin 95] in advance. The corpus is then divided into two parts; 90% of the sentences (i.e., 20,715 sentences, 169,237 words) are used as the training corpus, and the remaining 10% (2,301 sentences, 19,030 words) are used as the testing set.

According to our experience in machine translation, most interested English compounds are of length 2 or 3. Longer compounds only constitute a small fraction of interested compounds; and such long compounds can be extended by slightly modifying the definition for some association metrics. Hence, only bigrams and trigrams compounds are investigated in the current work. The corpus is therefore scanned from left to right with the window sizes 2 and 3. The lists of bigrams and trigrams thus acquired then form the lists of compound candidates of interest.

All bigrams and trigrams are submitted to three independent lexicographers of a local MT-based service translation center. The lexicographers inspect all n-grams and decide which n-grams should be considered as compounds and entered into the compound dictionary for the MT system. When there is inconsistency among their choices, the lexicographers will negotiate for a compromise. The final candidates are then used as the standard for evaluating the performance of the proposed compound extraction method. Since all the bigrams and trigrams are scanned for qualification before any experiment is conducted, the performance will reasonably reflect the performance against human judgement, the criterion for including an n-gram or not will thus not be biased by the algorithm designer's intention to have high performance. In contrast to other evaluation methods, which may submit the output of the classifier to the lexicographers for evaluation, the current evaluation process will therefore be more objective.

The parameters for the compound model Λ_c and the non-compound model $\Lambda_{\bar{c}}$ are evaluated from the above-mentioned training corpus, which is tagged with parts of speech and normalized into stem forms. The n-grams in the training corpus are further divided into two classes. The compound class comprises the n-grams in the compound dictionary, which was constructed by the lexicographers as described above; and the non-compound class consists of the remaining n-grams which are not in the compound dictionary. However, n-grams that occur only once or twice are excluded from consideration because such n-grams rarely introduce inconsistency and the estimated feature values are highly unreliable.

For each class, the means and standard deviations of the mutual information, normalized frequency, dice metric, contextual entropy and POS discrimination are estimated. The outlier entries (outside the range of 3 standard deviations from the mean) are discarded before estimating the model parameters so that the estimated parameters are more robust.

2.9 Baseline Models

To achieve minimum error classification, several factors must be carefully considered, including the features to be used, the model for formulating the underlying probability density functions of the two-classes, and the estimation to the parameters of the density functions. In the simplest form, only one feature is used for classification. We then have the following baseline models:

$$\lambda = \frac{f(X_i|\mathbf{C})P(\mathbf{C})}{f(X_i|\overline{\mathbf{C}})P(\overline{\mathbf{C}})}$$

where X_i refers to any of the features among normalized frequency (NF), mutual information (MI), dice metric (D), contextual entropy (H) and POS discrimination (Dpos). In the baseline models, the probability density function is modeled with a normal distribution. Such baseline models are used to evaluate the performance for the individual feature; they will also be compared with other more complicated models in later sections to justify our proposals.

Table 1 gives the performance using only one feature, where Dpos, MI, H, NF and D refer to the POS discrimination, mutual information, contextual entropy, normalized frequency and dice, respectively. The shaded areas highlight the error rate performance, which is the optimization criterion at the current stage. The features are arranged in increasing order of error rates for bigrams.

		Training Set					Testing Set				
Feature		Dpos	MI	H	NF	D	Dpos	MI	H	NF	D
Bigram Baseline	Recall	11.09	0.0	4.87	6.01	12.33	8.07	0.0	1.35	2.69	36.77
	Precision	100.0	*	30.92	30.69	37.07	100.0	*	23.08	33.33	57.75
	Error Rate	11.03	12.41	13.15	13.34	13.47	21.20	23.06	23.78	23.68	20.79
	WPR(1:1)	55.54	*	17.90	18.35	24.70	54.03	*	12.22	18.01	47.26
	F-measure	19.97	*	8.41	10.05	18.50	14.93	*	2.55	4.98	44.93
Feature		Dpos	MI	H	NF	D	Dpos	MI	H	NF	D
Trigram Baseline	Recall	0.0	0.0	13.99	10.20	7.58	0.0	0.0	12.07	3.45	39.66
	Precision	*	*	42.11	22.58	25.49	*	*	58.33	66.67	41.07
	Error Rate	4.95	4.95	5.21	6.18	5.67	11.51	11.51	11.11	11.31	13.49
	WPR(1:1)	*	*	28.05	16.39	16.54	*	*	35.20	35.06	40.37
	F-measure	*	*	21.00	14.05	11.69	*	*	20.00	6.56	40.35

Table 1 Error rate performance using only one feature
(*: undefined, i.e., all candidates are classified as non-compound.).

The error rates are in the ranges of 11.03%-13.47% and 20.79%-23.78% for bigrams in the training set and the testing set respectively; for trigrams the error rates are in the ranges of 4.95%-6.18% (training set) and 11.11%-13.49% (testing set); such performance corresponds to accuracy rates of 87-89% (76-79%) and 94-95% (87-89%) in classifying the bigram and the trigram training (testing) set. Using the minimum error classifier thus achieves moderately low error rates both for the training set and the testing set, without resorting to arbitrary thresholding.

Initially, however, the precision and recall are not sufficiently high except for the bigram POS discrimination case since the classifier tends to recognize most n-grams (or even all n-grams) as non-compounds. The 0% recalls and undefined precisions (designated as '*') in the table are the results of classifying all entries as non-compound as suggested by the assumed normal distributions. Such initial precisions and recalls are not a critical problem at the current stage where minimization of error counts is the major goal. It will be shown in later sections that, by incorporating more features, the error rates will be further reduced and the precision and recall will be indirectly improved toward high precision and moderate recall. By applying a learning algorithm for optimizing precision and recall at the second stage, such performance indices will be improved further so that appropriate precision and recall could be achieved while retaining a low error rate.

There are several problems to achieve the minimum error criterion by using the above baseline models. First of all, various features are not used jointly to supplement each other so as to reduce the error rate. Second, the distributions are not necessarily normal for some features. (For instance, the normalized frequency is more likely to have an exponential distribution. Fortunately, the comparison between the baseline models and other more complicated models in the current work will not be affected significantly, since we actually get almost the same error rates for such features by using the exponential distribution assumption.) To resolve such problems, we will propose some methods in the following sections to improve the error rate performance further so that the first optimization stage is better conducted.

2.10 Integration of the Multiple Features with Automatic Feature Selection

While each of the above features provides moderately good initial error rate performance in the above baseline models, it is known that jointly considering all the features would, in general, achieve better performance. It is also known that step-by-step filtering approaches, which were commonly used in traditional extraction tasks, tend to raise the precision rate at the cost of lowering the recall, since a filtering module may filter out potential candidates without using all available information; it is then not likely to acquire the global optimal precision and recall achievable by using such features. Using all features jointly in one step for optimizing the extraction task is thus emphasized here.

However, increasing the number of features may increase the modeling complexity of the classifier [Devijver 82] without increasing much performance, since some of the features might be highly correlated, and thus much redundant information will be contained in the whole set of features. Therefore, an automatic mechanism for choosing the right features is proposed here, so that only a subset of the most discriminative features are used for efficient computation without losing much discrimination power.

Since our goal is to minimize the error rate performance at the current optimization stage, our strategy for finding the best feature set is to combine the current feature set (which is initially empty) with each feature not in the current feature set for conducting the likelihood ratio test. The feature which enables the classifier to minimize the error rate performance, when jointly considered with the current set of optimal features, is then added to the optimal set of features. This process starts from the baseline models and stopped when the inclusion of new feature do not improve the training set performance further. This strategy can be characterized as a kind of sequential forward selection (SFS) in the literature [Devijver 82].

2.11 Optimization Using Independent Normal Model

The performance of the classifier will also depend on how good the density function of the features fits the real training data. In the simplest model, the joint probability of the features is approximated as the product of the probabilities for the individual features (by assuming that they are statistically independent), and each feature is assumed to be normally distributed. The corresponding log-likelihood ratio then becomes:

$$\log\lambda = \sum_{i=1}^{D}\left[\log f\left(x_i|\mathbf{C}\right) - \log f\left(x_i|\overline{\mathbf{C}}\right)\right] + \left[\log P(\mathbf{C}) - \log P\left(\overline{\mathbf{C}}\right)\right]$$

where the summation is taken over all features being used, and D is the dimension of the feature vector. In other words, all features are assumed to be independent in such a simplified model. With such assumptions, uncorrelated (complementary) features are likely to be included earlier than highly correlated features since features with smaller correlation coefficients tend to be closer to the independent assumption and are likely to have better performance (because features that are independent normally suggests more extra information). The mechanism can thus select the most useful and complementary features automatically and leave redundant features unused. Table 2 shows the performances for using different numbers of features, which are selected, in sequence, by the automatic feature selection method described in the previous section, using independent normal assumption.

By applying the feature selection mechanism over all the features, the Dpos (discrimination), H (entropy), MI (mutual information), NF (normalized frequency) and D (dice) features are selected in sequence for bigrams; on the other hand, the best feature sequence for trigrams, under the current model, is Dpos, MI, H, D, NF. The SFS strategy results in the following error rate performance, where the features are arranged in the same order as the sequence in the feature selection process. For instance, the second column of the bigram performance table shows that the error rate is 8.07% when the entropy feature, H, is added to the feature set with other preceding features (in this case, the discrimination feature, Dpos).

		Training Set					Testing Set				
Feature Sequence		Dpos	H	MI	NF	D	Dpos	H	MI	NF	D
bigram	Recall	11.09	40.41	54.61	35.34	31.30	8.07	35.43	60.54	33.63	50.67
	Precision	100.0	88.04	77.39	71.04	49.67	100.0	89.77	92.47	82.42	66.47
	Error Rate	11.03	8.07	7.61	9.81	12.46	21.20	15.82	10.24	16.96	17.27
	WPR(1:1)	55.54	64.23	66.00	53.19	40.49	54.04	62.60	76.51	58.03	58.57
	F-measure	19.97	55.39	64.03	47.20	38.40	14.93	50.81	73.17	47.77	57.50
Feature Sequence		Dpos	MI	H	D	NF	Dpos	MI	H	D	NF
trigram	Recall	0.0	14.29	33.53	29.45	26.24	0.0	17.24	44.83	56.90	48.28
	Precision	*	100.0	70.99	46.98	33.83	*	100.0	86.67	49.25	47.46
	Error Rate	4.95	4.24	3.97	5.14	6.19	11.51	9.52	7.14	11.71	12.10
	WPR(1:1)	*	57.15	52.26	38.22	30.04	*	58.62	65.75	53.08	47.87
	F-measure	*	25.01	45.55	36.20	29.56	*	29.41	59.09	52.80	47.86

Table 2 Error rate performances of the independent normal model.

The shaded areas highlight the error rate performance, which is the optimization criterion at the current stage. The parts of speech discrimination is selected first in the two feature sequences, since the parts of speech information provide the best error rate performance among all using the normal assumption. For the bigram case, the error rate is reduced by 26.8% (from 11.03% to 8.07) when the contextual entropy information, H, is included. The inclusion of the mutual information further reduces the error rate performance to 7.61%, corresponding to a reduction of 5.7% of the remaining errors. For trigrams, the error rates are improved slightly from 4.95% to 4.24% to 3.97 when the second and the third features (i.e., MI and H) are included, corresponding to the error reduction rates of 14% and 6%, respectively.

With the improvement in error rate performance, the extra features improve the precision and recall performance (WPR or FM, or both) as well. Although the error rate is only slightly improved (and the system retains essentially the same low error rates), the precision and recall performance is shifted away from the initial low precesion and recall status significantly.

However, it fails to further improve the error rate performance as the feature dimension increases further, since the statistically independent assumption for the joint density function becomes harder and harder to be true as the feature dimension increases. For instance, the dice metric (D) and mutual information (MI) has a high correlation coefficient of about 0.6 in bigrams and 0.4 in trigrams. Another example would be the NF (normalized frequency) and H (contextual entropy), which have correlation coefficients of about 0.4-0.5 in the bigram and trigram data. The problem is resolved by considering the feature correlation and using better density functions to approximate the joint distribution as follows.

2.12 Model Refinement with Mixtures of Gaussian Density Functions

There are two sources of errors for including new features in the previous model, which assumes that the features are *statistically independent* and *normally distributed*. First, the independent assumption might not be true for some feature pairs. In fact, the correlation coefficients for the features indicate that some of the features are highly correlated. Therefore, it is desirable to use a multivariate Gaussian (i.e., normal) distribution [Roussas 73, Rabiner 93], which encode feature correlations with a covariance matrix, to consider the effects of the correlations among the features. Second, the distributions of some features cannot be approximated by a normal distribution. Therefore, using a mixture of the multivariate normal distribution would be a better way to fit the density functions. By increasing the number of mixtures, it is possible, intuitively, to fit the shapes of the real distributions better, and thus have better estimation on the likelihoods of the joint feature vectors. In other words, we have [Roussas 73]:

$$f(\mathbf{x}|\mathbf{C}) \equiv \sum_{i=1}^{K} r_i \cdot N\left(\mathbf{x};\mu_\mathbf{i},\Sigma_\mathbf{i}\right), \quad \sum_{i=1}^{K} r_i = 1$$

$$N(\mathbf{x};\mu,\Sigma) = (2\pi)^{-D/2}|\Sigma|^{-1/2}\exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\mu)\right]$$

where $f(\mathbf{x}|\mathbf{C})$ is the density function of the feature vector $\mathbf{x}$ for the compound-class, K is the number of mixtures, r_i is the prior probability for the *i*-th mixture, $N\left(\mathbf{x};\mu_\mathbf{i},\Sigma_\mathbf{i}\right)$ is the multivariate Gaussian density function for the *i*-th mixtures, $\mu_\mathbf{i}$ is the mean vector, $\Sigma_\mathbf{i}$ is the covariance matrix for the *i*-th mixture, and D is the number of features used (i.e., the dimension of features) in the feature vector $\mathbf{x}$. By using such a density function, the features will be jointly considered for decision making, and the correlation between the two features are taken into account in terms of the covariance matrices.

2.12.1 Using Multivariate Gaussian with Fixed Number of Mixtures

To fit the training data into a mixture of multivariate Gaussian distribution, we must estimate the number of mixtures, K, the means and co-variances of each mixture or cluster. The clusters are acquired using a standard unsupervised estimation technique [Duda 73, Schalkoff 92]. For a K-mixture distribution, the feature vectors are clustered into K clusters; the mean vectors and covariance matrices and prior probabilities of the clusters are then estimated from the clustering results. The procedure is outlined in the following algorithm.

[Estimation of Mean Vectors and Covariance Matrices of the Gaussian Mixtures]

Goal: To split n data into K clusters

Initialization: Regard all feature vectors in the training set $Y = \{y_1, y_2, \ldots, y_n\}$ as one cluster, C_1, and evaluate its mean vector μ_1.

Iteration: for q=2, K (K: number of mixtures) do Steps 1-5, which create the new clusters C_1 through C_q, as follows.

Step1: Select a cluster which satisfies a splitting criterion (currently, maximum determinant is used), and split the mean μ of the selected cluster into two means $\mu_1 = \mu + \delta\mu$ and $\mu_2 = \mu - \delta\mu$, where δ is a small constant for shifting the original means in two opposite directions.

Step2: Re-assign data y_i, i=1, 2, …, n to cluster C_j if

$$j = \underset{k}{\operatorname{argmax}}\, N(y_i;\ \mu_k,\ \Sigma_k),$$

Step3: Re-Calculating the mean vectors μ_j, and covariance matrix Σ_j, j=1,…, q for C_1, ..., C_q.

Step4: Repeat Steps 2-4 until the mean vectors and covariance matrices converge.

Step5: Go to Step 1 for creating a new cluster, until we have K clusters.

The above procedure is a K-mean [Duda 73] clustering process which tries to approximate the maximum likelihood estimate of the parameters of a multivariate Gaussian mixture density function

[Duda 73, Rabiner 93]. The algorithm for obtaining the maximum likelihood estimate of the parameters of the mixture density functions, based on the EM algorithm [Dempster 77], can be found in [Duda 73]. It is possible to increase the likelihood of the training data by conducting EM estimation on the parameters using the formula in [Duda 73]. However, it requires more computation time (about 3 or 4 times longer) and results in only slightly improved error rate performance (as shown in some of our tests), we will therefore use the current K-mean algorithm throughout this dissertation.

Since the number of mixtures for the underlying distributions of the joint features of various dimensions are not known, we first fixed the number of mixtures (K) throughout the whole feature selection process to find the best performance. The cases for fixing K=1, 2, 3 are tried in order to find the best number of mixtures to use. The results for bigrams and trigrams are given in the Tables 3-5.

		Training Set					Testing Set				
Feature Sequence		Dpos	H	MI	D	NF	Dpos	H	MI	D	NF
bigram	Recall	11.09	41.04	49.33	43.73	32.12	8.07	36.32	52.91	54.71	44.39
	Precision	100.0	89.39	77.52	68.06	56.78	100.0	90.00	90.77	71.35	68.28
	Error Rate	11.03	7.92	8.06	9.53	11.45	21.20	15.62	12.10	15.51	17.58
	WPR(1:1)	55.54	65.21	63.43	55.90	44.45	54.03	63.16	71.84	63.03	56.34
	F-measure	19.97	56.25	60.29	53.25	41.03	14.93	51.75	66.85	61.93	53.80
Feature Sequence		Dpos	H	MI	D	NF	Dpos	H	MI	D	NF
trigram	Recall	0.0	25.66	32.07	27.70	24.49	0.0	34.48	44.83	46.55	44.83
	Precision	*	67.18	66.27	49.48	33.60	*	86.96	89.66	47.37	48.15
	Error Rate	4.95	4.30	4.17	4.98	6.13	11.51	8.13	6.94	12.10	11.90
	WPR(1:1)	*	46.42	49.17	38.59	29.04	*	60.72	67.24	46.96	46.49
	F-measure	*	37.14	43.22	35.52	28.34	*	49.38	59.78	46.96	46.43

Table 3 Error rate performance using a 1-mixture multivariate Gaussian density function (K=1).

		Training Set					Testing Set				
Feature Sequence		Dpos	H	MI	NF	D	Dpos	H	MI	NF	D
bigram	Recall	69.84	71.50	71.61	50.67	51.71	69.06	71.30	69.96	67.26	47.09
	Precision	100.0	97.87	88.93	62.93	45.53	100.0	95.78	93.41	80.65	52.24
	Error Rate	3.74	3.73	4.63	9.82	13.67	7.14	7.34	8.07	11.27	22.13
	WPR(1:1)	84.92	84.69	80.27	56.80	48.62	84.53	83.54	81.68	73.95	49.66
	F-measure	82.24	82.63	79.34	56.14	48.42	81.70	81.75	80.00	73.34	49.53
Feature Sequence		Dpos	MI	H	NF	D	Dpos	MI	H	NF	D
trigram	Recall	54.22	58.31	62.97	39.94	55.98	56.90	62.07	70.69	51.72	43.10
	Precision	100.0	100.0	86.06	48.41	28.79	100.0	100.0	95.35	71.43	19.69
	Error Rate	2.27	2.06	2.34	5.08	9.04	4.96	4.37	3.77	7.94	26.79
	WPR(1:1)	77.11	79.15	74.51	44.18	42.38	78.45	81.03	83.02	61.58	31.39
	F-measure	70.32	73.67	72.73	43.77	38.02	72.53	76.60	81.19	60.00	27.03

Table 4 Error rate performance using a 2-mixture multivariate Gaussian density function (K=2).

		Training Set					Testing Set				
Feature Sequence		Dpos	NF	MI	H	D	Dpos	NF	MI	H	D
bigram	Recall	68.91	71.30	72.12	67.05	61.76	69.06	71.30	70.40	65.92	53.36
	Precision	100.0	97.18	90.74	83.70	50.81	100.0	97.55	94.01	93.63	73.46
	Error Rate	3.86	3.82	4.37	5.71	12.16	7.14	7.03	7.86	8.89	15.20
	WPR(1:1)	84.46	84.24	81.43	75.37	56.29	84.53	84.42	82.21	79.77	63.41
	F-measure	81.59	82.25	80.37	74.46	55.75	81.70	82.38	80.51	77.37	61.82
Feature Sequence		Dpos	H	MI	D	NF	Dpos	H	MI	D	NF
trigram	Recall	63.27	68.22	67.06	51.90	54.23	75.86	74.14	74.14	36.21	37.93
	Precision	100.0	95.12	90.91	80.91	39.08	100.0	97.73	95.56	95.45	41.51
	Error Rate	1.82	1.75	1.96	2.99	6.45	2.78	3.17	3.37	7.54	13.29
	WPR(1:1)	81.63	81.67	78.98	66.40	46.65	87.93	85.93	84.85	65.83	39.72
	F-measure	77.50	79.45	77.18	63.24	45.43	86.27	84.32	83.50	52.50	39.64

Table 5 Error rate performance using a 3-mixture multivariate Gaussian density function (K=3).

For bigram compound word detection, the best (training set) error rate performance is found in Table 4 when Dpos (parts of speech discrimination) and H (contextual entropy) are used jointly using a 2-mixture multivariate (i.e., bivariate) Gaussian density function. The best feature sequence is identical to the independent normal model. In this case, the error rate, 3.73%, is only about 49% of the best independent normal model (using Dpos, H and MI), whose error rate is 7.61%. The WPR is also significantly improved from 66.00 to 84.69, and the FM from 64.03 to 82.63. The precision and recall for this case are 97.87% and 71.50%, respectively.

Trigram compound detection also acquires the best results by using Dpos and H, but with a 3-mixture multivariate Gaussian density function (Table 5). The error rate is 1.75% in this case, which is only 44% of its counterpart using the independent normal model, i.e., 3.97% (using Dpos, MI and I). The results demonstrates that using a mixture of multivariate Gaussian density function to include the correlation and fit the density function of the training data does reduce the error rate and improve the precision, recall, WPR and FM significantly. The comparison between the independent normal model and the K-mixture multivariate Gaussian model (using fixed K throughout the feature selection process) is summarized in Table 6.

N		Training Set					Testing Set				
	Model & Features	P	R	E	WPR	FM	P	R	E	WPR	FM
2	IN: Dpos+H	88.04	40.41	8.07	64.23	55.39	89.77	35.43	15.82	62.60	50.81
	IN: Dpos+H+MI	77.39	54.61	7.61	66.00	64.03	92.47	60.54	10.24	76.51	73.17
	Mx: Dpos+H (K=2)	97.87	71.50	3.73	84.69	82.63	95.78	71.30	7.34	83.54	81.75
3	IN: Dpos+MI	100.0	14.29	4.24	57.15	25.01	100.0	17.24	9.52	58.62	29.41
	IN: Dpos+MI+H	70.99	33.53	3.97	52.26	45.55	86.67	44.83	7.14	65.75	59.09
	Mx: Dpos+H (K=3)	95.12	68.22	1.75	81.67	79.45	97.73	74.14	3.17	85.93	84.32

Table 6 Comparison between Independent Normal (IN) Model and K-mixture Multivariate Gaussian (Mx) Model. (2: bigram, 3: trigram, P: Precision, R: Recall, E: Error Rate, WPR: Weighted Precision/Recall with equal weights, FM: F-measure.)

Again, the WPR and FM are, in general, improved when the error rate is reduced. However, those tables indicate that the error rates do not decrease monotonically as the number of features are increased for a given K; the error rate decrease only for the first two or three features in the feature sequence. Besides, the error rates do not decrease monotonically either when the number of mixtures increased when comparing the performance for a specific number of features. There are several possibilities which make the fitting of the training data to a K-mixture D-variate density function imperfect in the above process; the performance thus is not monotonically increased with K or D.

First of all, the number of mixtures for the underlying density function of the joint features may not be characterized by a small K when the number of features increases to some extent. The number of mixtures is usually a function of the number of features. In fact, it is known in statistical pattern recognition community [Devijver 82] that when the number of features increases, the best number of mixtures for modeling the joint distribution of the features, in general, will increase quickly. For instance, two features, each having two Gaussian mixtures, when considered jointly, may have as many as four mixtures if they are independently distributed. With correlation considered, the number of mixtures maybe slightly less than four mixtures; but the number still grows exponentially with the number of features in the worst cases. As a result, the real K may far exceed our searching range (K=1-3) when new features are included.

Second, even though the underlying density function is a K*- mixture Gaussian density function, it does not mean that the maximum likelihood estimation algorithm using a larger K (K>K*) always produces a better density function. The reason is: the estimation algorithm adopted here for estimating the mixture density (or more complicated estimation algorithms like EM), in general, could only achieve a local maximum of the likelihood function. Therefore, even if we use a larger K, the estimation algorithm may still not converge at the global maximum likelihood, and thus may not always have better performance than using the density function acquired by using a smaller K.

For instance, if the real number of mixtures of the density function is exactly 2, then using a 3-mixture approximation to fit the 2-mixture data, theoretically, can at least achieve the same maximum likelihood estimate as the ideal 2-mixture estimate (in which one of the three mixtures virtually does not exist). However, the estimation process for the 3-mixture case may be trapped at a local maximum of the likelihood function, such that its likelihood value is less than the one acquired by using 2 mixtures.

Therefore, increasing the number of mixtures does not guarantee to achieve a better maximum likelihood estimate for the training data; furthermore, since the maximum likelihood criterion in estimating the density functions does not imply minimum error rate performance, increasing the number of mixtures in an attempt to improve the error rate performance may not be achievable as expected.

Therefore, increasing K does not necessarily improve the error rate performance monotonically, especially when K is small. However, increasing K in general will still increase the likelihood values of the training data, since most real data cannot be perfectly fitted with a small number of mixtures, and the best K tends to be large. The above arguments might suggest that we should use different K for different number of features, and search for the best mixture density by trying various K's that are less than a constant Kmax.

Third, even though the number of mixture is known precisely, the estimated parameters (mean vectors and covariance matrices) acquired from the Gaussian mixture estimation algorithm may not really maximize the likelihood of the training data [Duda 73]; in most cases, only a local maximum of the likelihood function is acquired and there may have multiple local maxima which have the same likelihood values [Duda 73]. In the estimation process, how the (local) maximum likelihood estimate converge depends on the selection of the cluster to split in the splitting step. When the number of features increase, the best number of mixtures increase rapidly, a better initial guess is thus harder to locate. Therefore, increasing the number of features has the additional problem that a better local maximum is harder to find. All these factors contribute to the non-monotonic performance curve for increasing K.

2.12.2 Improvement by Searching for the Best Number of Mixtures

A simple way to overcome the above described problems in using a fixed number of mixtures *throughout* the whole feature selection process is to set an upper bound, Kmax, and tries all K <= Kmax *during* the feature selection process for each feature dimension. We thus tries several Kmax and find the best K (denoted by K*) for such searching ranges. Table 7 shows the results when Kmax=3. In this table, the numbers in the parentheses indicate the best number of mixtures (K*) used for all the features up to the current column. For instance, the Dpos(.)-H(2) feature sequence in Table 7 for bigrams means that a local optimal is found when Dpos and H are jointly considered using a 2-mixture density function. Similarly, the feature sequence Dpos(.)-H(.)-MI()-NF()-D(1) indicates that the best performance for jointly considering the five features [Dpos, H, MI, NF, D] is to use a 1-mixture density function, instead of using 2 or 3 mixtures.

		Training Set					Testing Set				
Feature Sequence		Dpos(2)	H(2)	MI(3)	NF(3)	D(1)	Dpos	H	MI	NF	D
bigram	Recall	69.84	71.50	72.12	67.05	32.12	69.06	71.30	70.40	65.92	44.39
	Precision	100.0	97.87	90.74	83.70	56.78	100.0	95.78	94.01	93.63	68.28
	Error Rate	3.74	3.73	4.37	5.71	11.45	7.14	7.34	7.86	8.89	17.58
	WPR(1:1)	84.92	84.69	81.43	75.37	44.45	84.53	83.54	82.21	79.77	56.34
	F-measure	82.24	82.63	80.37	74.46	41.03	81.70	81.75	80.51	77.37	53.80
Feature Sequence		Dpos(3)	H(3)	MI(3)	D(3)	NF(1)	Dpos	H	MI	D	NF
trigram	Recall	63.27	68.22	67.06	51.90	24.49	75.86	74.14	74.14	36.21	44.83
	Precision	100.0	95.12	90.91	80.91	33.60	100.0	97.73	95.56	95.45	48.15
	Error Rate	1.82	1.75	1.96	2.99	6.13	2.78	3.17	3.37	7.54	11.90
	WPR(1:1)	81.63	81.67	78.98	66.40	29.04	87.93	85.93	84.85	65.83	46.49
	F-measure	77.51	79.45	77.19	63.24	28.34	86.27	84.32	83.50	52.50	46.43

Table 7 Error rate performance using multivariate Gaussian density function up to 3 mixtures (Kmax=3).

The training set error rates in Table 7 demonstrates that, by searching for the best K in [1, Kmax] for each feature dimension, the training set error rate performance is always better than (or identical to) its counterpart in Tables 3-5 of the same number of features. This justify our arguments that K should be searched in a given range, instead of using a fixed number of mixtures all the time, for better performance.

Table 7, however, still do not show monotonic decreasing of the error rates when the number of features are increased. In fact, the error rates no more decrease after the third feature is included, just like Tables 3-5. The problem is that Kmax=3 is still too small to search for a better performance even with only 3 features. To partially justify this argument, cases for Kmax=4,5 for bigram extraction are listed as follows.

		Training Set					Testing Set				
Feature Sequence		Dpos(2)	H(2)	MI(4)	NF(4)	D(1)	Dpos	H	MI	NF	D
bigram	Recall	69.84	71.50	73.99	67.25	32.12	69.06	71.30	73.54	66.37	44.39
	Precision	100.0	97.87	92.61	83.63	56.78	100.0	95.78	96.47	95.48	68.28
	Error Rate	3.74	3.73	3.96	5.69	11.45	7.14	7.34	6.72	8.48	17.58
	WPR(1:1)	84.92	84.69	83.30	75.44	44.45	84.53	83.54	85.01	80.93	56.34
	F-measure	82.24	82.63	82.26	74.55	41.03	81.70	81.75	83.45	78.30	53.80

Table 8 Error rate performance for bigrams using multivariate Gaussian density function up to 4 mixtures (Kmax=4).

		Training Set					Testing Set				
Feature Sequence		Dpos(2)	H(5)	MI(4)	NF(5)	D(5)	Dpos	H	MI	NF	D
bigram	Recall	69.84	70.78	73.06	67.15	57.51	69.06	70.85	74.44	67.71	56.05
	Precision	100.0	100.0	94.89	85.82	56.98	100.0	100.0	95.40	96.79	63.45
	Error Rate	3.74	3.63	3.83	5.45	10.66	7.14	6.72	6.72	7.96	17.58
	WPR(1:1)	84.92	85.39	83.97	76.49	57.25	84.53	85.43	84.92	82.25	59.75
	F-measure	82.24	82.89	82.56	75.34	57.24	81.70	82.93	83.63	79.68	59.52

Table 9 Error rate performance for bigrams using multivariate Gaussian density function up to 5 mixtures (Kmax=5).

Those tables demonstrates that the training set error rates for any given number of features do decrease monotonically as Kmax is increased (and the best K, K*, thus found are also increased). We can thus expected that the error rate will decrease monotonically as the number of features is increased if we allow a much larger searching ranges. In the current task, however, it is obvious that the best number of mixtures, even for 3 features, are more than ten (i.e., K*>10). This would requires a very lengthy time to converge. Furthermore, the features at the tail of the feature sequence are highly correlated with features at the front of the list, which means that they may only provide little additional information once those features selected earlier are used for classification. Improving the estimation of the density functions for including such features thus would not likely to produce significant improvement. Therefore, compromise must be taken between modeling complexity and computation costs.

With Kmax=3 and two features (in which the training set error rates are minimal), we actually have testing set WPR performance of 84% (bigram) and 86% (trigram); the F-measures are about

82% and 84% for bigram and trigram, respectively.

Note that, since the training set n-grams and the testing set n-grams are sampled from the same domain, and there might be overlapped n-grams in the two sets, the repoted testing set performance in the above tables might be a little more optimistic than the performance by evaluating on n-grams of a completely different domain. Such general domain-dependence and robustness issues, however, are beyond the scope of the current research; therefore, they will not be studied as separated topics in the current work.

Since error rate improvement does not show great changes when Kmax is above 3 for 2 features and above, we will therefore take Kmax=3 and two features in subsequent discussions in order to investigate other ways for optimization.

Given the above error rate performance, it is still possible to further improve the error rate performance by better estimation methods or learning methods, and thus indirectly improve the precision and recall rate performance. However, such approaches do not guarantee to get the best joint precision-recall performance, since the minimum error rate criterion, eventually, is not equivalent to the maximum precision-recall criterion for a given precision-recall function. After the error rate is reasonably reduced, we are interested in optimizing the precision and recall performances by adjusting the parameters of the classifier. Therefore, the following chapter will highlight such optimization techniques and leave the other methods for improving the error rate performance to future works.

Chapter 3
Precision and Recall Optimization by Nonlinear Parameter Learning

In the previous chapter, the problems for acquiring good precision-recall performance is obtained indirectly through a minimum error classifier. Designing problems for achieving minimum classifications error are addressed, including the integration of multiple features and the consideration to take feature correlations into account. In this chapter, the learning techniques for optimizing a user-specified criterion function of precision and recall is discussed, so that the criterion function is maximized directly.

The optimization technique proposed here is essentially a gradient descent method for moving toward the best WPR (or FM) performance. The idea is to define a risk function associated with the optimization criterion, and approximate the risk as a differentiable function of the system parameters. When an input token is misclassified, the parameters which are used in classifying the misclassified instance are adjusted in the reverse direction of the gradient vector of the risk function in the parameter space, where the gradient vector is derived by differentiating the risk function with respect to the various parameters. Since the optimization function is defined in terms of precision and recall, which could be evaluated only after all input tokens are classified, the adjustments are actually conducted in batch after all training instances are classified.

The learning rules for each parameters depends on the optimization criterion to optimize. Since the learning process for optimizing FM is almost identical to the learning process for WPR optimization, only the latter is described in the following sections. The detail learning rules for FM optimization are given in Appendix I.

3.1 Adaptive Learning for Optimizing Weighted Precision and Recall

Let the precision rate and the recall rate be p and r, respectively, the risk function, R, of this particular classification result for the optimal WPR criterion could be defined as

$$
\begin{aligned}
R(p, r; \Lambda, \{\mathbf{x}\}) &\triangleq w_p(1-p) + w_r(1-r) \\
w_p &= \text{weighting factor for } p \\
w_r &= \text{weighting factor for } r
\end{aligned}
$$

which is, obviously, a function of the system parameters, Λ, for the two classes and the set of input feature vectors, $\{\mathbf{x}\}$, associated with the training data. The parameters include the means and variances of the density functions as well as the log probabilities for the POS patterns for the two classes.

Our goal is to optimize this function by changing the parameters Λ of the discrimination functions. Such adjustment is possible if we know the relationship between the risk and the parameters. To find the optimal set of parameters, in the sense of minimum risk as measured by the weighted precision and recall, we can find the gradient $\nabla R(\Lambda)$ of the risk $R(\Lambda)$ and adjust the parameters in the reverse direction, that is, let

$$
\begin{aligned}
\vec{\delta}_{\Lambda}(t) &= -\varepsilon(t)\nabla R / \|\nabla R\| \\
\Lambda(t+1) &= \Lambda(t) + \vec{\delta}_{\Lambda}(t)
\end{aligned}
$$

where t is the iteration index, $\vec{\delta}_{\Lambda}(t)$ is the adjustment to the parameter vector at the t-th iteration, $\varepsilon(t)$ is a linear time-decreasing function which decreases the amount of adjustments with time, $\|\nabla R\|$ is the Euclidean norm of the gradient vector of the risk function, $\Lambda(t)$ and $\Lambda(t+1)$ are the parameter vectors before and after learning, respectively, and the ∇ operator contains partial derivative operators with respect to the system parameters; by adjusting the parameters in this way, non-increasing change of average risk $\delta\overline{R}$ in the direction of $\vec{\delta}_{\Lambda}$ can be expected if the adjustment is sufficiently small, because

$$
\delta\overline{R} \approx \nabla R \bullet \vec{\delta}_{\Lambda}(t) = -\varepsilon \nabla R \bullet \nabla R / \|\nabla R\| \leq 0
$$

This kind of learning is generally known as the gradient descent method [Amari 67, Su 91, Juang 92]. The learning rule for each parameter depends on the optimization criterion to optimize. The detailed learning rules for WPR is given in the following sections.

3.2 Learning Rules for Optimizing Weighted Precision-Recall and F-Measure

Let $n_{\alpha\beta}$ be the number of class-α input tokens which are classified as class-β (α, β = 1 for compound, and 2 for non-compound, respectively). Furthermore, let n_1 represent the number of compounds in the compound class, which is a constant. We then have the precision *p* and recall *r* as follows:

$$p = \frac{n_{11}}{n_{11}+n_{21}}$$

$$r = \frac{n_{11}}{n_{11}+n_{12}} = \frac{n_{11}}{n_1}$$

Since the weighted precision and recall is defined as

$$WPR = w_p * p + w_r * r \qquad (w_p + w_r = 1),$$

to maximize WPR is therefore equivalent to minimizing the following risk function.

$$\begin{aligned} R &= w_p(1-p) + w_r(1-r) \\ &= w_p \frac{n_{21}}{n_{11}+n_{21}} + w_r \frac{n_{12}}{n_{11}+n_{12}} \\ &= w_p \frac{n_{21}}{n_1 - n_{12} + n_{21}} + w_r \frac{n_{12}}{n_1} \end{aligned}$$

The above error counts n_{12} and n_{21} are determined by the discrimination function $g(\mathbf{x})$ of the classifier, which classifies an input vector $\mathbf{x}$ as class-1 if $g(\mathbf{x}) \geq 0$; otherwise, the input is classified as class-2. Obviously, $n_{\alpha\beta}$ depend on the parameters, Λ, used by the discrimination function, and the input training patterns $\{\mathbf{x}\}$. Therefore, the risk function can be expressed in terms of the system parameters; and, we can adjust the system parameters in the reverse direction of ∇R to optimize the risk.

To evaluate ∇R, the error counts must be replaced with some differentiable functions. The error counts n_{12} and n_{21}, in the current task, is approximated with the differentiable loss functions l_{12} and l_{21} as follows. Assuming that $c(\mathbf{x})$ refers to the real class label of the input feature vector, and the class label assigned by the classifier is determined by the discrimination function $g(\mathbf{x})$, then l_{12} and l_{21} can be defined as:

$$
\begin{aligned}
n_{12} &\approx l_{12} \equiv \sum_{\mathbf{x}:c(\mathbf{x})=1,g(\vec{\mathbf{x}})<0} l(-d_{\mathbf{x}}) \\
n_{21} &\approx l_{21} \equiv \sum_{\mathbf{x}:c(\mathbf{x})=2,g(\vec{\mathbf{x}})\geq 0} l(+d_{\mathbf{x}})
\end{aligned}
$$

where $l(d_{\mathbf{x}})$ is an approximation to a zero-one loss function which approximates the integer number (=1) to be added to the error counts when a classification error is made, and $d_{\mathbf{x}}$ is a misclassification distance which reflects how bad the decision was made according to a given discrimination function $g(\mathbf{x})$. (Both $d_{\mathbf{x}}$ and $g(\mathbf{x})$ will be defined later.) When a mistake is made, and $d_{\mathbf{x}}$ is large, the loss should be close to 1; otherwise, it should be close to zero. Such a loss function can be defined as follows [Amari 67]:

$$
l(d_{\mathbf{x}}) = \frac{1}{\pi}\tan^{-1}\left(\frac{d_{\mathbf{x}}}{d_0}\right) + \frac{1}{2}
$$

where the shape of the arctan function is controlled by the constant d_0, and the distance function $d_{\mathbf{x}}$ is propotional to $g(\mathbf{x})$ (as described later).

For simplicity, the likelihood ratio for the independent normal model is used to illustrate the learning procedure, that is (Section 2.11):

$$
\log\lambda = \sum_{i=1}^{D}[\log f(x_i|\mathbf{C}) - \log f(x_i|\overline{\mathbf{C}})] + [\log P(\mathbf{C}) - \log P(\overline{\mathbf{C}})] .
$$

Then the discrimination function can be defined as a weighted version of the log-likelihood ratio:

$$
\begin{aligned}
\mathrm{g}(\mathbf{x},\ \mathbf{w},\ \Lambda) &= \sum_{i=1}^{D} w_i[\log\ f(x_i|\mathbf{C}) - \log\ f(x_i|\overline{\mathbf{C}})] + w_0[\log P(\mathbf{C}) - \log P(\overline{\mathbf{C}})] \\
&\equiv \mathbf{w}\bullet\mathbf{s} \\
\mathbf{s} &= [\ldots,\ \log\ f(x_i|\mathbf{C}) - \log f(x_i|\overline{\mathbf{C}}),\ldots,\ \log P(\mathbf{C}) - \log P(\overline{\mathbf{C}})] \\
&= [\ s_1,\ s_2,\ldots,s_D,s_0] \\
\mathbf{w} &= [w_1,w_2,\ldots,w_D,w_0]
\end{aligned}
$$

where $\mathbf{w}$ is a weighting vector of D+1 dimensions for the individual log-likelihood scores (i.e.,

$s_i \equiv \log f\left(x_i|\mathbf{C}\right) - \log f\left(x_i|\overline{\mathbf{C}}\right)$ (i=1, D) and $s_0 \equiv \log P(\mathbf{C}) - \log P\left(\overline{\mathbf{C}}\right)$) in the log likelihood ratio, and Λ is the parameter vector.

The weights are introduced here for normalizing the individual log-likelihood scores so that the discrimination function is not biased by any particular term that has a large dynamic ranges. Initially, all weights are set to 1. With the above discrimination function, the misclassification distance $d_{\mathbf{x}}$ can be defined as a normalized version of the discrimination function:

$$d_{\mathbf{x}} = \frac{g(\mathbf{x}, \mathbf{w}, \Lambda)}{\|\mathbf{w}\|} = \frac{\mathbf{w} \bullet \mathbf{s}}{\|\mathbf{w}\|}$$

Given the differentiable version of the loss functions, we have:

$$\begin{aligned}
\nabla R &\approx w_p \nabla \frac{l_{21}}{n_1 - l_{12} + l_{21}} + w_r \nabla \frac{l_{12}}{n_1} \qquad (n_{12} \approx l_{12}, \quad n_{21} \approx l_{21}) \\
&= w_p \frac{\left(n_1 - l_{12} + l_{21}\right)\nabla l_{21} - l_{21}\nabla\left(n_1 - l_{12} + l_{21}\right)}{\left(n_1 - l_{12} + l_{21}\right)^2} + w_r \nabla \frac{l_{12}}{n_1} \\
&\approx w_p \frac{\left(n_1 - n_{12}\right)\nabla l_{21} + n_{21}\nabla l_{12}}{\left(n_{11} + n_{21}\right)^2} + w_r \nabla \frac{l_{12}}{n_1} \qquad (l_{12} \approx n_{12}, \; l_{21} \approx n_{21}) \\
&= \frac{w_p n_{11}}{\left(n_{11} + n_{21}\right)^2} \nabla l_{21} + \left[\frac{w_p n_{21}}{\left(n_{11} + n_{21}\right)^2} + \frac{w_r}{n_1}\right] \nabla l_{12} \\
&\equiv k_{21} \nabla l_{21} + k_{12} \nabla l_{12}
\end{aligned}$$

where

$$\begin{aligned}
k_{21} &= \frac{w_p n_{11}}{\left(n_{11} + n_{21}\right)^2} \\
k_{12} &= \frac{w_p n_{21}}{\left(n_{11} + n_{21}\right)^2} + \frac{w_r}{n_1}
\end{aligned}$$

Since

$$\begin{aligned}
\nabla l_{12} &= \sum_{\mathbf{x}: c(\mathbf{x}) = 1, g(\mathbf{x}) < 0} \nabla l\left(-d_{\mathbf{x}}\right) = -\sum l'\left(-d_{\mathbf{x}}\right) \nabla d_{\mathbf{x}} \\
\nabla l_{21} &= \sum_{\mathbf{x}: c(\mathbf{x}) = 2, g(\vec{\mathbf{x}}) \geq 0} \nabla l\left(+d_{\mathbf{x}}\right) = +\sum l'\left(+d_{\mathbf{x}}\right) \nabla d_{\mathbf{x}}
\end{aligned}$$

and $l(d)'$ is symmetric about 0, i.e.,

$$l'(+d_{\mathbf{x}}) = \frac{1}{\pi}\frac{d_0}{d_{\mathbf{x}}^2 + d_0^2} = l'(-d_{\mathbf{x}})$$

the gradient of the risk function is equivalent to

$$\nabla R \approx k_{21}\left[\sum_{\substack{c(\mathbf{x})=2\\ g(\mathbf{x})\geq 0}} l'(d_{\mathbf{x}})\nabla d_{\mathbf{x}}\right] - k_{12}\left[\sum_{\substack{c(\mathbf{x})=1\\ g(\mathbf{x})<0}} l'(d_{\mathbf{x}})\nabla d_{\mathbf{x}}\right]$$

This means to accumulate the gradient of the loss ($l'(d_{\mathbf{x}})\nabla d_{\mathbf{x}}$) into a vector for each class-2 tokens that is misclassified as class-1, and accumulate the gradient of the loss into another vector, if a class-1 input token is misclassified into class-2. When all misclassified instances are inspected, and their gradient of loss be estimated, the gradient of the risk function is approximated as the weighting sum of the two vectors, weighted by k_{21} and k_{12}. The parameters are then adjusted by $-\varepsilon(t)\nabla R/\|\nabla R\|$.

The gradient of the distance function $\nabla d_{\mathbf{x}}$ is evaluated by differentiating the distance function with respect to the system parameters. If the parameters to be adjusted is the weighting factors w_i, it is easy to show that

$$\nabla d_{\mathbf{x}} = \frac{\mathbf{s}}{\|\mathbf{w}\|} - \frac{g(\mathbf{x})\mathbf{w}}{\|\mathbf{w}\|^3}$$

If the parameters of the probability density functions (i.e., the means and variances of the probability density functions) are to be adjusted, ∇ would be equal to $\sum_{ijk}\mathbf{u_{ijk}}\frac{\partial}{\partial\theta_{ijk}}$, where θ_{ijk} refers to the jth parameter (i.e., mean or variance) corresponding to the ith feature x_i (among NF, MI, D, H, Dpos), for the kth class (k=1, 2), and $\mathbf{u_{ijk}}$ corresponds to the unit vector for the θ_{ijk} component, and

$$\nabla d_{\mathbf{x}} = \frac{1}{\|\mathbf{w}\|}\sum_i w_i \sum_{jk}\mathbf{u_{ijk}}\frac{\partial}{\partial\theta_{ijk}}\left[\log f(x_i|\mathbf{C}) - \log f(x_i|\overline{\mathbf{C}})\right]$$

In case the discrete POS feature is used, the above factors $\log f(x_i|\mathbf{C})$ will be replaced with log probabilities of the form $\log P_{ij}\left(C_j | C_i\right)$, where C_i, C_j are the POS's at position i and j of an extended POS pattern. Since such log probabilities are, by themselves, the parameters to be adjusted, the gradient with respect to the POS-related parameters is simply

$$\nabla d_{\mathbf{x}} = \frac{1}{\|\mathbf{w}\|} \sum_i w_i \sum_{jk} \mathbf{u_{ijk}} (-1)^{k+1} \qquad k = 1 \text{ or } 2$$

Because the gradient operator is additive, the three types of parameters can be adjusted individually or simultaneously using the corresponding $\nabla d_{\mathbf{x}}$.

For instance, if the mean, $\mu_{MI,C}$, of the mutual information feature (*m*) for the compound class is to be adjusted for improving the WPR performance, the learning rule for this parameter will be

$$\mu_{MI,C}(t+1) = \mu_{MI,C}(t) - \varepsilon(t) \frac{\partial R}{\partial \mu_{MI,C}} / \|\nabla R\|$$

where the component of the gradient vector of the risk function for this parameter is

$$\frac{\partial R}{\partial \mu_{MI,C}}(t) = k_{21} \sum_{\substack{c(\mathbf{x}) = 2 \\ g(\mathbf{x}) \geq 0}} \frac{1}{\pi} \frac{d_0}{d_{\mathbf{x}}^2 + d_0^2} \frac{m - \mu_{MI,C}(t)}{\sigma_{MI,C}^2(t)} - k_{12} \sum_{\substack{c(\mathbf{x}) = 1 \\ g(\mathbf{x}) < 0}} \frac{1}{\pi} \frac{d_0}{d_{\mathbf{x}}^2 + d_0^2} \frac{m - \mu_{MI,C}(t)}{\sigma_{MI,C}^2(t)}$$

assuming that the mutual information is independent with other features and the variance of the normal distribution is $\sigma_{MI,C}^2$. (The time indices *t* and *t+1* refer to the values at the current and next iterations, respectively.) Since the two constants k_{21} and k_{12} are determined by the error counts, which could be determined only after all misclassification instances are inspected, the update action is actually conducted in batch (i.e., in 'training by epoch' mode [Schalkoff 92]) after all input are classified, instead of updating the parameters on each misclassification. In other words, as the risk function, R, is a nonlinear function of the loss function of individual error tokens, conventional 'training by sample' approaches [Amari 67, Juang 92, Schalkoff 92] frequently used in the literatures can no longer be applied. Nonlinear learning is therefore required, and is derived in this dissertation.

The derivation procedure for updating the parameters to optimize the F-measure criterion is quite similar to the above procedure for WPR optimization. The details are given in Appendix I. In fact, the form of the learning rules turns out to be the same as the learning rules for WPR optimization. The only difference is the two constants k_{21} and k_{12}, which control the amount of adjustment for the two different types of classification errors. As given in the Appendix, for F-measure optimization, we have

$$k_{21} = \frac{1}{(\beta^2+1)} \times \frac{1}{(n_1 - n_{12})} \qquad k_{12} = \frac{1}{(\beta^2+1)} \times \frac{(\beta^2 n_1 + n_{21})}{(n_1 - n_{12})^2}.$$

3.3 Performance after Learning on Independent Normal Model

The above learning algorithms for optimizing WPR and FM can be applied to any discrimination functions whose density functions are differentiable with respect to their parameters. The following sections demonstrate the effect of the learning algorithms on the simplest model, namely the independent normal (IN) model, and the more complicated K-mixture multivariate Gaussian density functions (whose best K (K*) is searched below Kmax=3) with the best two features.

Table 10 illustrates the changes of the WPR and FM performance when the parameters of the normal density function are adjusted to optimize the WPR and FM criteria directly according to the learning rules of the previous sections. The Wp and Wr are both set to 0.5, i.e., (Wp:Wr)=(1:1), and the β argument for FM is set to 1, so that there is no particular preference for precision or recall is emphasized. The performance using the first two features selected by the SFS mechanism with an independent normal model (denoted by 'IN: Dpos+H') are repeated in the table for comparison. To highlight the changes in WPR and FM, the performance before learning and the performance after learning are shaded in the WPR and FM columns.

		Training Set					Testing Set				
	Model	P	R	E	WPR	FM	P	R	E	WPR	FM
bi-gram	IN: Dpos+H	88.04	40.41	8.07	64.23	55.39	89.77	35.43	15.82	62.60	50.81
	Mx:Dpos+H(Kmax=3)	97.87	71.50	3.73	84.69	82.63	95.78	71.30	7.34	83.54	81.75
	IN+LRN: WPR (1:1)	97.35	72.44	3.66	84.89	83.07	97.56	71.75	6.93	84.65	82.69
	IN+LRN: FM	94.67	73.68	3.78	84.18	82.87	96.41	72.20	7.03	84.31	82.56
tri-gram	IN: Dpos+MI	100.0	14.29	4.24	57.15	25.01	100.0	17.24	9.52	58.62	29.41
	Mx:Dpos+H(Kmax=3)	95.12	68.22	1.75	81.67	79.45	97.73	74.14	3.17	85.93	84.32
	IN+LRN: WPR (1:1)	98.24	65.01	1.79	81.63	78.24	100.0	74.14	2.98	87.07	85.15
	IN+LRN: FM	98.23	64.72	1.80	81.48	78.03	97.73	74.14	3.17	85.94	84.31

Table 10 Learning results on Independent Normal Model
(IN: Independent Normal Model, Mx: Mixture of Multivariate Gaussian Model,
IN+LRN: Learning on Parameters of Independent Normal Model.)

The upper part of Table 10 shows the performance for bigram compound word extraction for independent normal (IN) model, and the learning results acquired by applying the learning algorithm on the IN model (IN+LRN). In comparison with the IN model, the learning algorithm for WPR optimization significantly improves the WPR performance by 20.66% (from 64.23% to 84.89%) in the training set; and the improvement in the testing set is 22.05% (from 62.60% to 84.65%). The precision and recall in the testing set are improved from 89.77 to 97.56% (+7.79%) and from 35.43 to 71.75% (+36.32%), respectively. In fact, the FM is also improved by more than 27% in both the training set and testing set. Such performance enable the classifier to include most of the compounds into the list and the lexicographer would acquire the compounds from such a list easily. The results for optimizing F-measure by learning, on the other hand, is improved by 27.48% (=82.87-55.39) in the training set and 31.75% (=82.56-50.81) in the testing set. The testing set precision and recall converged at 96.41% precision and 72.20% recall, respectively, with this optimization criterion.

For trigram compound extraction, the learning algorithm for WPR optimization improves the WPR performance by 24.48% (=81.63-57.15%) and 28.45% (=87.07-58.62%) in the training set and the testing set, respectively. For FM optimization, the improvement is 53.02% (78.03-25.01%) and 54.90% (84.31%-29.41%) for the training set and the testing set, respectively.

The effect demonstrates clearly that the precision and recall performance could be improved directly toward different optimization criteria by applying the proposed learning technique. Such improvement is comparable with the improvement using the more complicated K-mixture multivariate Gaussian density functions. In fact, as shown in Table 10, the learning results are slightly better than the case for using two features with Kmax=3 without learning; this situation is changed only when a larger Kmax is used, such as Kmax=5 (as shown in Table 9). It also has the additional advantages that the model is easy to implement and the computation is relatively more efficient in comparison with the mixture density model, which requires very long time to converge for a large Kmax. However, using the mixture model would provide a better initial error rate performance for conducting the learning process, as shown in the following sections.

3.4 Performance after Learning on Gaussian Mixture Model

The learning algorithm could also be applied to the results acquired by the K-mixture multivariate Gaussian model, which was used to improve the error rate performance of the independent normal model. This would provide the learning algorithm a better initial error rate performance, so that precision and recall optimization get better results. We thus start with the parameters estimated by the K-mixture multivariate Gaussian model, then adjust the parameters to optimize the WPR and FM. The results are given in Table 11.

	Model	Training Set P	R	E	WPR	FM	Testing Set P	R	E	WPR	FM
bi-gram	IN: Dpos+H	88.04	40.41	8.07	64.23	55.39	89.77	35.43	15.82	62.60	50.81
	Mx:Dpos+H(Kmax=3)	97.87	71.50	3.73	84.69	82.63	95.78	71.30	7.34	83.54	81.75
	IN+LRN: WPR (1:1)	97.35	72.44	3.66	84.89	83.07	97.56	71.75	6.93	84.65	82.69
	IN+LRN: FM	94.67	73.68	3.78	84.18	82.87	96.41	72.20	7.03	84.31	82.56
	Mx+LRN: WPR (1:1)	99.57	72.75	3.42	86.16	84.07	100.0	71.75	6.52	85.87	83.55
	Mx+LRN: FM	99.43	72.85	3.42	86.14	84.09	100.0	71.75	6.52	85.87	83.55
tri-gram	IN: Dpos+MI	100.0	14.29	4.24	57.15	25.01	100.0	17.24	9.52	58.62	29.41
	Mx:Dpos+H(Kmax=3)	95.12	68.22	1.75	81.67	79.45	97.73	74.14	3.17	85.93	84.32
	IN+LRN: WPR (1:1)	98.24	65.01	1.79	81.63	78.24	100.0	74.14	2.98	87.07	85.15
	IN+LRN: FM	98.23	64.72	1.80	81.48	78.03	97.73	74.14	3.17	85.94	84.31
	Mx+LRN: WPR (1:1)	99.25	76.68	1.18	87.96	86.51	100.0	75.86	2.78	87.93	86.27
	Mx+LRN: FM	98.14	76.97	1.21	87.56	86.27	100.0	75.86	2.78	87.93	86.27

Table 11 Learning Results on Mixture of Multivariate Gaussian Model
(IN: Independent Normal Model, Mx: Mixture of Multivariate Gaussian Model, IN+LRN: Learning on Independent Normal Model. Mx+LRN: Learning on Multivariate Gaussian Mixtures)

In these cases, the optimization method improves the WPR and FM for bigrams by 1.47% and 1.46%, respectively, in the training set, resulting in 2.33% improvement in WPR and 1.80% improvement in FM for the testing set data. For trigrams, the training (testing) set performance is improved by 6.29% (2.00%) WPR and 6.82% (1.95%) FM, respectively.

The results show that, with a better initial error rate performance, the WPR or FM performance could be better improved in general when comparing the learning results between the independent normal model and the multivariate Gaussian mixture model. This is particularly clear in the trigram cases where the training set WPR is further improved by 6.33% and FM by 8.24% in comparison with the results by applying the learning rules on simple independent normal model. The improvement in bigram cases are less impressive and the changes are only 1.27% WPR and 1.22% FM, respectively, with respect to the learning results using an independent normal model. The extra improvement in testing set for bigram data are 1.22% WPR and 0.99% FM; and the additional improvement for trigram data are 0.86% WPR and 1.96%FM.

The above results also indicate that a simple model will result in a worse error rate performance in the first stage and make a larger jump in the (second) optimization stage. A more complicated model, on the other hand, will have a better initial error rate performance and the jump in the optimization stage is smaller. The combination of these two stages, however, prefers a lower error rate in the first stage before the precision and recall optimization techniques are applied.

3.5 Tailoring to Users' Preference with the Learning Mechanism

To show that the learning mechanism do follow a user's preference over a particular precision-*vs*-recall profile, the weights for precision and recall for optimizing WPR are set to different values. For FM optimization, different β's are given to specify different preference. The following table shows how the system parameters are adjusted to enhance precision and/or recall.

	Training Set					Testing Set				
Model	P	R	E	WPR	FM(β)	P	R	E	WPR	FM
Mx:Dpos+H(Kmax=3) before learning	97.87	71.50	3.73	78.10 (1:3) 84.69 (1:1) 91.28 (3:1)	91.15 (0.5) 82.63 (1.0) 75.58 (2.0)	95.78	71.30	7.34	77.42 83.54 89.66	89.63 81.75 75.14
Mx+LRN: WPR (1:3)	94.08	74.09	3.79	79.09 (1:3)	82.90	97.58	72.20	6.83	78.54	82.99
Mx+LRN: WPR (1:1)	99.57	72.75	3.42	86.16 (1:1)	84.07	100.0	71.75	6.52	85.87	83.55
Mx+LRN: WPR (3:1)	99.71	72.02	3.50	92.79 (3:1)	83.63	100.0	71.30	6.62	92.83	83.25
Mx+LRN: FM(0.5)	99.57	72.75	3.42	86.16	92.73(0.5)	100.0	71.75	6.52	85.87	92.70
Mx+LRN: FM(1.0)	99.43	72.85	3.42	86.14	84.09(1.0)	100.0	71.75	6.52	85.87	83.55
Mx+LRN: FM(2.0)	89.51	75.13	4.18	82.32	77.62(2.0)	97.02	73.09	6.72	85.06	76.89

Table 12 Bigram Learning Results for Different User Preferences over Precision and Recall (WPR(1:3): Wp=0.25 & Wr=0.75, WPR(1:1): Wp=0.5 & Wr=0.5, WPR(3:1): Wp=0.75 & Wr=0.25, FM(β), for β=0.5, 1, 2 are given sequentially.)

The three WPR rates before learning, as shown in the 3rd row of the above table, correspond to the performance for the weights (Wp:Wr)=(1:3), (1:1) and (3:1), respectively. The three FM's on the same row indicate FM(0.5), FM(1.0) and FM(2.0), respectively, for the given precision and recall as rated by different lexicographers (or β's).

It is clear, from the above table, that the system parameters are adjusted so that WPR and FM are improved for all the three pairs of (Wp:Wr) weights and the three different β's. For instance, WPR(1:3) for training set is improved from 78.10% to 79.09%, and FM(0.5) for training data is improved from 91.15% to 92.73% after applying the learning rules. Therefore, the learning mechanism successfully improve the above six instances of optimization criteria (possibly from six different lexicographers) as expected.

In addition, the recall will be emphasized and the precision be de-emphasized when we have a higher Wr than a Wp in those three instances of WPR optimization criteria. When Wp is given a higher weight, the learning will converge at a local optimum which has a higher precision than that of the (1:1) case, and a lower recall than that of the (1:1) case. This fact can be found from the non-decreasing precision rates highlighted in the ‘P’ column of the above table, and the non-increasing recall rates in the ‘R’ column of the above table.

For the three FM optimization criteria, the non-increasing precision rates and non-decreasing recalls for the increasing β’s also follow our expectation as explained in Section 2.1 exactly. That is, the precision is lowered and recall increased for user who specify a higher β. This demonstrates our expectation for optimizing different user preference or criteria automatically.

3.6 Learning Strategies for Integrating Discrete POS Feature

Many useful features for terminology extraction are discrete-valued in nature. The discrete POS feature as mentioned in Section 2.7 is one such feature. It is therefore interesting to combine such features with continuous-valued word association features as discussed in the previous sections. Unlike the continuous valued features, which are characterized with only a few parameters such as distributional mean vectors and covariance matrices, the discrete-valued features have a much larger number of tunable parameters. Therefore, some issues must be carefully handled when combining these kinds of features. In particular, the dynamic ranges of the discrete feature and other continuous features may be different to a large extent. Therefore, it might be necessary to weight them differently when combining these two different types of features. In this way, the system performance will not be dominated by a particular type of features, and the system can thus really enjoy the joint information provided by all features. The following sections address such problems by exercising the integration of the discrete POS feature with the other continuous-valued word association features (i.e., mutual information, normalized frequency, dice, mutual information and POS discrimination.)

3.6.1 Combining Discrete POS Feature and Continuous Word Association Features

The performance for using only the part-of-speech likelihood feature is shown in the following table. It shows a larger training set error rate than the other continuous-valued features as shown in Table 1 but with a smaller testing set error rate performance.

	Training Set	Testing Set
Bi-gram Recall	89.33	84.75
Bi-gram Precision	38.47	61.76
Bi-gram Error Rate	19.05	15.62
WPR (1:1)	63.90	73.26
FM	53.77	71.46

Table 13 Error rate performance using only POS likelihood.

To combine POS likelihood with other features, we include the extra POS likelihood score in the log-likelihood ratio during the sequential forward selection process, and select the best features, among the continuous-valued features, in sequence. The results for using up to 3 mixtures in the multivariate Gaussian mixture density functions for the continuous-valued features are shown in the following table.

		Training Set					Testing Set				
Feature Sequence		Dpos(3)	D(3)	H(3)	NF(1)	MI(1)	Dpos	D	H	NF	MI
bigram	Recall	92.54	92.12	92.85	87.67	87.97	89.69	89.69	90.58	86.55	86.55
	Precision	43.20	45.85	52.99	43.12	43.12	70.42	66.01	61.59	67.48	67.72
	Error Rate	16.02	14.47	11.11	15.88	15.89	11.07	13.03	15.20	12.72	12.62
	WPR(1:1)	67.87	68.99	72.92	65.39	65.55	80.05	77.85	76.08	77.01	77.13
	F-measure	58.90	61.23	67.47	57.81	57.87	78.90	76.05	73.32	75.84	75.99

Table 14 Error rate for combining POS likelihood with other continuous-valued features (with Kmax=3) (The numbers to the right of the features are the best number of mixtures for the joint features up to the current feature in the feature sequence.)

The error rate performance is gradually improved when new features are included until the feature dimension exceeds 3. (See the effect of increasing the number of features in Section 2.12.1

for the non-monotonicity behavior here.)The best training set error rate reach a minimum of about 11% with such integration. Although such integration improves the error rate performance in comparison with the POS likelihood only model, the error rate is still high and the WPR or FM performance are still less than satisfactory. In particular, the minimum error rate is a little higher than using continuous-valued features alone with the simple independent normal model (but with higher WPR and FM), as shown in Table 2.

The WPR and FM could be further improved by applying the learning rules on the features. For instance, the results by using POS likelihood and the first two continuous-valued features in the preferred feature sequence and by applying the learning rules on the continuous features show such improvement clearly in the following table. By learning on the continuous-valued features, we actually get a model that is better than the independent normal model.

Bigram	Training Set					Testing Set				
Model	P	R	E	WPR	FM	P	R	E	WPR	FM
Mx:Dpos+D(Kmax=3)	45.85	92.12	14.47	68.99	61.23	66.01	89.69	13.03	77.85	76.05
Mx+LRN	75.90	78.65	5.75	77.28	77.25	81.19	79.37	9.00	80.28	80.27

Table 15 Bigram performance by combining POS likelihood and other features and by learning parameter values on continuous-valued features.

3.6.2 Dealing with Different Dynamic Ranges among Features

Although the WPR and FM are improved to an acceptable level and the error rates are indirectly improved in these cases, such integration still does not outperform the performance given in previous sections. The fact that the simple combination of the discrete-valued feature with the continuous-valued features do not have an impressive performance indicates that these two kind features are not weighted appropriately. One problem is that we are not considering the different dynamic ranges of the log-probabilities of these two kinds of features, and we are not considering the relative contributions of the two kinds of features for discrimination. Before applying the learning rules, we therefore seek to improve the error rate performance by weighting the two kinds of features differently.

To handle the different dynamic ranges we assign a weight, Wpos, to the POS likelihood score and another weight to the log-likelihood ratio of the other continuous-valued features. The weights are adjusted starting from the unweighted case (i.e., Wpos=0.5) so as to minimize the error rate of the training set data. The best weight thus acquired is 0.100644 for the present case. By multiplying Wpos to the POS likelihood score and (1-Wpos) to the log-likelihood score of the other features, the results in the 4th row of Table 16 are acquired.

Bigram	Training Set					Testing Set				
Model	P	R	E	WPR	FM	P	R	E	WPR	FM
Mx:Dpos+D(Kmax=3)	45.85	92.12	14.47	68.99	61.23	66.01	89.69	13.03	77.85	76.05
Mx:Dpos+D (+Wpos)	80.33	75.34	5.35	77.83	77.75	86.24	73.09	8.89	79.67	79.13

Table 16 Bigram performance by weighting POS likelihood and continuous features differently (with Wpos = 0.100644 for POS likelihood score and 1-Wpos for Dpos and D)

By integrating these two kinds of features in this simple way, the error rate performance is greatly improved even without learning. About 63% (i.e., (14.47-5.35)/14.47) error is removed after applying the weighting factors. Weighting these two kinds of features thus provide us a better initial error rate performance for WPR and FM optimization.

Note, however, that, the weights are based on the feature sequence which is determined by assuming that the POS likelihood feature is equally weighted against the other continuous-valued features. The best feature sequence might have changed after the weights are applied. We thus fix the weights and then re-select the features to further reduce the error rate. The new feature sequence, in this case, becomes: Dpos-H-D-MI-NF, instead of Dpos-D-H-NF-MI in the previous case, and the performance for using these two continuous-valued features are listed as follows.

Bigram	Training Set					Testing Set				
Model	P	R	E	WPR	FM	P	R	E	WPR	FM
Mx:Dpos+D (+Wpos)	80.33	75.34	5.35	77.83	77.75	86.24	73.09	8.89	79.67	79.13
Mx: PosL+Dpos+H	84.97	77.31	4.51	81.14	80.96	93.68	73.09	7.34	83.39	82.11

Table 17 Bigram performance by weighting POS likelihood and continuous features differently and then re-selecting features. (PosL: POS likelihood feature, Wpos=0.100644)

It is observed that 15.7% ((5.35-4.51)/5.35) of the remaining errors are removed with the new feature sequence when two continuous features are used in conjunction with the POS likelihood feature. It is possible to further reduce the error rates by re-adjust the weights and then re-selected the features repeatedly until the feature sequence converge. However, since the error rate is already very low, it is found that the weights will be changed only by about 0.4% by further adjusting the weights between the POS likelihood feature and other continuous-valued features, and the feature sequence converge quickly. In fact, the learning results for using the converged feature sequence is essentially the same as the results to be shown in the next section. Therefore, we will not go further in this direction; instead, optimization toward better precision and recall is conducted as illustrated in the following section.

3.6.3 Learning on Parts of Speech Likelihood Parameters

Under the error rate acquired by the weighted log-likelihood scores, we can further apply the learning rules on the parameters to optimize the WPR (or FM) performance. In the following tables, we show the results by using the first two continuous-valued features (Dpos and H) modelled with a 3-mixture density function together with the POS likelihood score. In this case, the parameters of the continuous-valued features are not adjusted in the learning process, instead, the parameters of the POS likelihood are adjusted to demonstrate that the learning technique is also applicable to discrete-valued features. In general, we should adjust the parameters of the continuous features jointly with that of the POS likelihood feature. However, it is found that the error rate may oscillate temporarily to the status where the error rate is more than 20%, and the converged error rate performance is only slightly better than that of fixing the parameters of the continuous features. Therefore, we choose not to go into such unstable status, which might converge to a worse local maximum; thus, the parameters of the POS likelihood feature, which are less likely to dramatically affect the decision boundary, are adjusted.

Bigram	Training Set					Testing Set				
Model	P	R	E	WPR	FM	P	R	E	WPR	FM
Mx: PosL+Dpos+H	84.97	77.31	4.51	81.14	80.96	93.68	73.09	7.34	83.39	82.11
Mx+PosLRN	92.00	81.04	3.23	86.52	86.17	97.78	78.92	5.27	88.35	87.34

Table 18 Bigram performance by learning POS likelihood (PosL) parameters
(after adjusting Wpos, and re-selecting features).

By applying the learning rules on the parameters of the POS likelihood feature, we are able to achieve the best WPR performance in comparison with the case where learning is applied on the mixtures of multivariate Gaussian density functions of continuous features. In fact, the testing set performance in WPR (88.35%) of the current model is also the best among all, which is 2.48% higher than its counterpart using mixtures of continuous features. Note that, since Wpos is only about 0.1, its contribution in the current task is limited by this weighting factor; in other tasks, which have a higher weight for discrete-valued features, like the POS likelihood feature in the current task, the above integrating might give more improvement.

3.7 Summary on Precision-Recall Maximization by Parameter Learning

Most simple mechanisms for terminology extraction rely on trial-and-error to setup empirical thresholds for each available features, and use such features to filter out inappropriate candidates step-by-step using one feature per step. Such simple thresholding-and-filtering approaches cannot automatically optimize a user specified criterion function of precision and recall. There is essentially no literature addressing the methods for optimizing the precision and recall directly. To resolve such optimization problems, a two-stage optimization scheme is proposed in this dissertation. In the first stage, the system tries to optimize the precision and recall performance indirectly through minimizing the classification error (by using a two-class classifier with a likelihood ratio test). In the second stage, an adaptive learning method is then applied to directly optimize a criterion function of precision and recall; such a criterion function can be pre-specified by a lexicographer based on the preference over the precision and recall performance. In particular, the optimal learning rules for optimizing WPR (weighted precision and recall) and FM (F-measure) are derived.

The method proposed here integrates mutual information, normalized frequency, dice, contextual entropy and part of speech (POS) information for discriminating compounds and non-compounds. The POS discrimination metric, in particular, is proposed in the current work for encoding the syntactic constrains over possible compound candidates. Syntactic constraints can thus be easily integrated quantitatively for jointly optimizing the system performance with other word association metrics.

To show the progressive improvement in each major step, the performance of various models for optimizing WPR in the bigram case are given in Table 19. The major optimization criteria, namely the error rate at the first stage and the WPR (and FM) at the second stage, are shaded in the table to highlight the progressive improvement.

Bigram	Training Set					Testing Set				
Model	P	R	E	WPR	FM	P	R	E	WPR	FM
IN: Dpos+H	88.04	40.41	8.07	64.23	55.39	89.77	35.43	15.82	62.60	50.81
Mx:Dpos+H(Kmax=3)	97.87	71.50	3.73	84.69	82.63	95.78	71.30	7.34	83.54	81.75
Mx+LRN: WPR (1:1)	99.57	72.75	3.42	86.16	84.07	100.0	71.75	6.52	85.87	83.55
Mx+ PosLRN	92.00	81.04	3.23	86.52	86.17	97.78	78.92	5.27	88.35	87.34

Table 19 Comparison of various models for bigram WPR optimization.
(IN: Independent Normal Model, Mx: Mixture of Multivariate Gaussian Model, Mx+LRN: Learning on Multivariate Gaussian Mixtures, Mx+PosLRN: learning POS likelihood (PosL) parameters (after adjusting Wpos, and re-selecting features))

In summary, the weighted recall and precision using the proposed approach can achieve as high as about 88% for bigram compounds and 88% for trigram compounds for a training (resp. testing) corpus of 20,715 (resp. 2,301) sentences sampled from technical manuals about cars. The F-measure performances are about 84% for bigrams and 86% for trigrams. By applying the proposed optimization method, the precision and recall profile is observed to follow the preferred criteria of different lexicographers.

Note that, although the two-stage optimization approach is proposed here, it is observed that the model with minimum error rate (after learning for minimizing error rate) generally produces satisfactory precision-recall performance. Hence, it is possible that some of the tasks in other domains may requires only the error rate minimization stage to get appropriate precision-recall performance. In this case, an alternative method is to use conventional learning methods for error rate minimization [Amari 67, Juang 92], and leave the nonlinear learning method proposed here as an option.

Chapter 4
Chinese Unknown Word Extraction

In the previous chapters, we had placed our emphases on the design of an optimal classifier, and a two-stage scheme to improve the joint precision-recall performance for extracting English compound words. In that task, the classifier is the only module in the system.

In this chapter, we will study how to extract Chinese unknown words from a text corpus. In this task, as described in Chapter 1, the classifier is not the only module. An extra segmentation module is usually required for segmenting the input sentences into words, before submitting potential new word candidates to the classifier for confirmation. The extraction task would then be conducted in a more complicated situation. For such a task, all available features in the two modules should be used, so that the best performance can be achieved. In the following chapters, we therefore are interested in developing techniques that improve both precision and recall by iteratively applying contextual constraints, provided by the segmentation module, and association features, used by the classifier module, to recover certain errors made in earlier iterations.

4.1 System Overview

In Chapter 1, the general problems of traditional Chinese unknown word extraction approaches are identified, and an iterative strategy was suggested for overcoming such problems. In the following sections, a brief system overview for the iterative scheme is presented first. We then discuss how unknown words introduce segmentation errors and how to improve such a problem by jointly including potential unknown words during the segmentation process. Techniques for refining the list of potential unknown words and the filter (or classifier) parameters are then addressed, so that we can iteratively improve the segmentation results and the filter parameters, and thus the output (unknown) word list. Performance evaluation is also conducted to estimate the system performance, with some significant tests to ensure that the improvement is statistically significant. Furthermore, quantitative analyses are given to justify our strategies, and segmentation errors are analyzed to find directions for our future works.

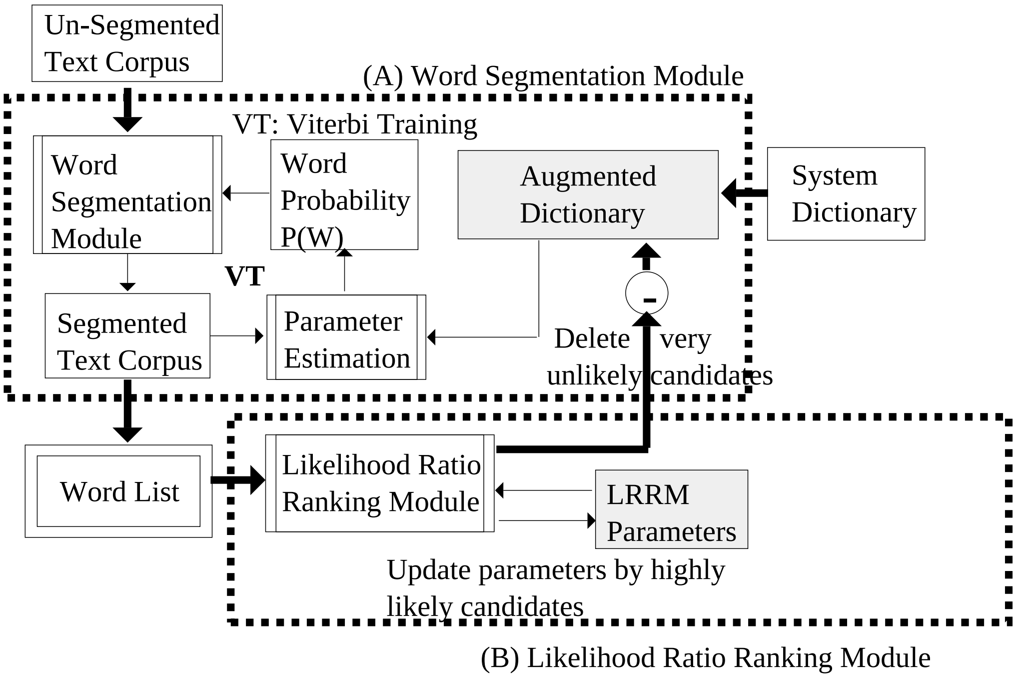

Figure 2 Chinese new lexicon identification system

Figure 2 illustrates the block diagram of the Chinese new lexicon identification system proposed in this dissertation. The dashed box A corresponds to the word segmentation module, and the 'filter' (i.e., the Likelihood Ratio Ranking Module) is represented by the dashed box B. The other blocks not in the dashed boxes are either input or output of the system.

Initially, an *augmented dictionary* is formed by combining the system dictionary and all the n-grams that occurs at least 5 times in the un-segmented input text corpus; such n-grams are the initial guess of the 'potential unknown words' and their word probabilities are estimated as their relative frequencies in the corpus. The initial probabilities are then used to segment the input corpus. Afterward, the word probabilities are re-estimated from the best segmentation pattern, which are explored using the augmented dictionary. The re-estimation process, called Viterbi Training [Rabiner 93], is then repeated until the segmentation patterns no more change or a specified number of segmentation iterations is reached.

The generated word list is then feed to the Likelihood Ratio Ranking Module (LRRM) in box B. A fraction of the word list entries which are judged as very unlikely to be words are truncated from the augmented dictionary. (The '-' sign in the circle means to truncate some very unlikely n-grams, ranked by the Likelihood Ratio Ranking Module, from the augmented dictionary.) In addition, a fraction of the word list entries which are judged as very likely to be words are used to update the parameter of the ranking module itself. Given the refined augmented dictionary, the Viterbi Training process is re-applied to find better segmentation pattern, and thus better output word lists. The process is then repeated so that the two modules can iteratively enhance the discrimination power of each other by progressively refining the augmented dictionary and the parameters of the ranking module, and thus improve the system performance.

The system performance is evaluated in terms of 'new word precision', 'new word recall' and the corresponding WPR and F-measure. Following our previous definitions for precision and recall, the new word precision, p, and the new word recall, r, are defined as follows:

$$p = \frac{\text{number of reported new words in the output list that are truly new words}}{\text{number of reported new words in the output list}}$$

$$r = \frac{\text{number of reported new words in the output list that are truly new words}}{\text{number of truly new words in the corpus}}$$

4.2 Experiment Environments and Evaluation Method

In our experiments, the un-segmented Chinese text corpus contains 311,591 sentences (about 1,670,000 words), which comes from the China Times Daily News. Since most Chinese words are less than 4 characters, only bigrams, trigrams and quadragrams (i.e., words of 2, 3 and 4 characters) are considered as word candidates in this study. Furthermore, only n-grams whose frequency is equal to or greater than 5 is considered as word candidates, since n-grams that rarely occur are considered less useful even though they are identified by the system. There are 242,042 distinct n-grams whose frequency of occurrence is equal to or greater than 5 in this corpus, including 99,407 bigrams, 99,211 trigrams and 43,424 quadragrams.

The system dictionary is a combination of the Academia Sinica dictionary [CKIP 90] and the BDC electronic dictionary [BDC 93] (excluding the words which never appear in the un-segmented text corpus, because such words will never be involved in the current task.) The merged dictionary contains 22,742 bigram words, 5,403 trigram words and 4,568 quadragram words, which sum up to 32,713 entries.

It is difficult to know exactly how many new words could be found in the large text corpus unless the corpus is manually inspected. Therefore, most lexicon extraction researches [Chiang 92, Lin 93b, Tung 94, Wang 95] did not report the recall rate. To give a reasonable estimate on the precision and recall rates, a sample corpus of 1,000 sentences is randomly sampled from the input text corpus. These sentences are segmented manually, by a linguistics department staff of Behavior Design Corporation, so that we are able to tell which n-grams in these samples are considered as words and which n-grams should be labelled as non-words. We then use the segmentation results generated from those sampled sentences to estimate the precision and recall of the system.

Although it is possible that different people may have different segmentation preference, various approaches in this dissertation, however, are compared against the same segmentation benchmark prepared by the same person. Therefore, the relative differences in performance among the various approaches will have small statistical variations, and are very likely to reflect the true situation. (The hypothesis testing, as given in Appendix II, had been conducted to ensure that the improvement of the proposed method is statistically significant. To estimate the various performance indices more precisely, though, we are planning to seek for a larger manually segmented corpus, such as the Academia Sinica Balance Corpus [Huang 95], for evaluating the system performance in our future works.)

This sample corpus contains 44,560 words. The numbers of distinct n-grams for n= 2, 3, 4 are 8,730, 9,658 and 8,745, respectively. Among those n-grams, 2,306 bigrams, 582 trigrams, and 394 quadragrams are recognized as distinct words in the manually segmented sample sentences. Among these words, 971 bigrams, 424 trigrams and 331 quadragrams are not registered in the system dictionary, and are thus considered as the new words in the sample corpus. However, since only those n-grams that occurs more frequent than 5 times in the whole input corpus are regarded as word candidates in the current task, only such candidates would be extracted by the system; therefore, the performance is estimated against such n-grams. In other word, by ‘new words’, we are actually referring to ‘new words that occurs at least 5 times in the input corpus’. Such ‘new words’ in the 1,000 sampled sentences include 866 bigrams, 295 trigrams, and 275 quadragrams, respectively; and these numbers are used as the ‘number of truly new words in the corpus’ for estimating new word precision/recall.

4.3 The Statistical Word Segmentation Model for Chinese New Lexicon Identification

Given an input Chinese text corpus, the (new) words in the text can be extracted by segmenting the input text into word segments first, hopefully that they are all correct, and then construct the (new) word list from the set of segments in the text corpus. Rule-based approaches [Ho 83, Chen 86, Yeh 91] as well as probabilistic approaches [Fan 88, Sproat 90, Chang 91, Chiang 92] for word segmentation had been proposed in the literature. Considering the capability of automatic training, adaptation to different domains, systematic improvement, and cost for maintenance, probabilistic approaches are more attractive for a large-scale system. Furthermore, probabilistic segmentation models were reported to be quite satisfactory [Chang 91, Chiang 92] when there is no unknown word in the corpus. Therefore, a probabilistic approach is adopted in this module.

Given a string of Chinese characters $c_1, c_2, \ldots, c_n$, represented as c_1^n, the best word segmentation pattern S^* (based on the *vocabulary* V), is defined as the segmentation pattern which has the maximal likelihood value among all possible segmentation patterns S_j:

$$S^*(V) = \underset{S_j}{\operatorname{argmax}} P\left(S_j = w_{j,1}^{j,m_j} | c_1^n,\ V\right)$$

where $w_{j,1}^{j,m_j} \equiv \left\{ w_{j,1}, w_{j,2}, \ldots, w_{j,m_j} \right\}$ are the concatenation of the m_j words in the j-th alternative segmentation pattern S_j, and V is the *vocabulary* of the system used in the segmentation process (i.e., the set of words being used in exploring various segmentation patterns). To make the estimation easier, the likelihood function is simplified as [Chang 91]:

$$P\left(S_j = w_{j,1}^{j,m_j} | c_1^n,\ V\right) \cong \prod_{i=1}^{m_j} P\left(w_{j,i} \mid V\right) \tag{1}$$

which, in spite of its simplicity, is shown to be effective in [Chiang 92] in comparison with other rule-based or statistical models. The *vocabulary* is explicitly included in the model to highlight the fact that the 'best' acquirable segmentation for an input corpus depends on the vocabulary, which is used to explore all possible segmentation patterns. Currently, our vocabulary corresponds to the set of n-grams in the augmented dictionary. Moreover, in unsupervised operation, the probabilities $P\left(w_{j,i} \mid V\right)$ are not known in advance, and must be estimated from the input corpus. We will use the maximum likelihood (ML) criteria as described later for this purpose. In the ML estimation process, the likelihood value for the corpus to consist of the selected sequence of word segments is maximal among all possible sequences. Since a word must be combined with other words to form the sentences, whether an n-gram will be identified as a word depends on its context; therefore, the segmentation process virtually imposes contextual constraints on the possible word sequence.

4.4 Viterbi Training for Parameter Re-estimation

The word probabilities in Equation (1) can be estimated from a large pre-segmented corpus, if it is available. However, such a training corpus is usually too expensive to construct. Therefore, an unsupervised training method, called Viterbi Training (VT) [Rabiner 93], is adopted in the current work for estimating the probabilities.

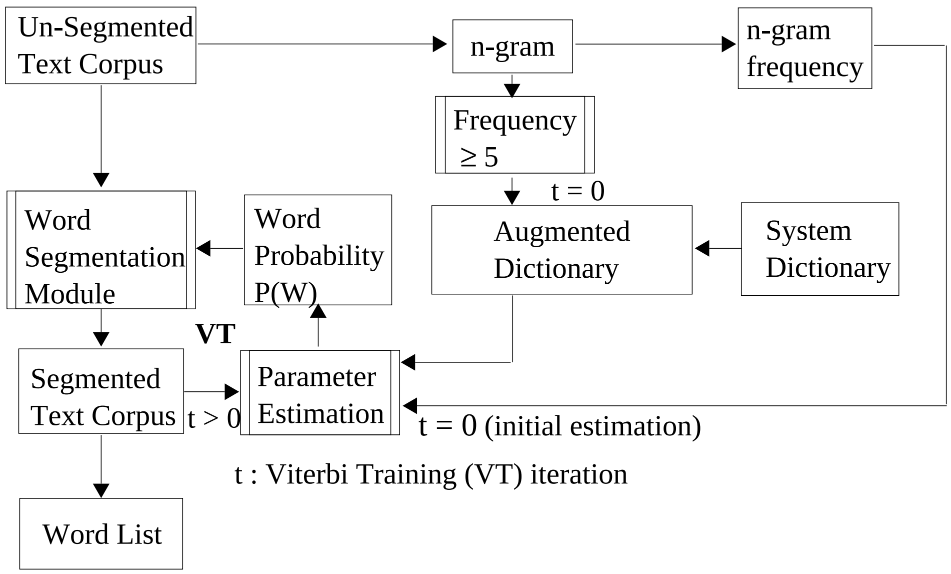

Figure 3 Viterbi training model for word segmentation using an augmented dictionary.

Figure 3 shows the block diagram of the word segmentation module, which applies the Viterbi training process to achieve the maximum likelihood segmentation of the corpus. The initial set of n-grams, which occurs at least 5 times in the input corpus, are acquired from the un-segmented text corpus with their frequencies. They are combined with the system dictionary words to form the initial augmented dictionary (i.e., the word candidates used for segmenting the corpus.) The initial probabilities of all the n-grams in the augmented dictionary are estimated as the relative frequencies of such n-grams in the input text corpus. Various segmentation patterns of the corpus are then explored in terms of the known words and unknown word candidates in the augmented dictionary. The path (i.e., the segmentation pattern) with the highest likelihood value is marked as the best path for the current iteration. A new set of parameters are then re-estimated based on the best path obtained by the Maximum Likelihood (ML) Estimator. This process repeats until the segmentation patterns no more change or a maximum number of iteration is reached. We then derive the word list from the segmented text corpus.

As indicated in Figure 3, only n-grams with at least 5 occurrences in the corpus will be included in the augmented dictionary; such restriction is intended to removes n-grams that are rarely used even though they might be judged as words. The reasons for applying the restriction are two folds. First, the statistics of such n-grams is usually too unreliable to use if their frequencies are less than 5. Second, there are a lot of n-grams which occurs only once or twice, most of which are insignificant. By removing such n-grams, the number of word candidates involved in the segmentation process (and thus the computation cost) can be greatly reduced. Note that the words in the system dictionary are always included in the augmented dictionary. Furthermore, all the single characters (unigrams) are included to model isolated characters.

4.5 Segmentation Using An Augmented Dictionary

As described earlier, the best acquirable segmentation depends on the vocabulary. The vocabulary used in the segmentation process, therefore, plays a very important role for acquiring the desired segmentation. If the vocabulary contains *all and only* the real words in the corpus, then the above model is very likely to generate the desired segmentation, and thus enable us to retrieve most real new words from the input corpus. On the contrary, if a large number of words in the text corpus is not included in the vocabulary, or the vocabulary contains many spurious words, then the segmented results will be far from ideal, and good performance for extracting new words cannot be expected. Therefore, if we are able to make good guess progressively on the vocabulary, then the unknown words are likely to be extracted with high recall and precision rates progressively.

However, most segmentation models (such as [Chang 91]) are based on the assumption that there is no unknown words in the text corpus. Hence, the unknown words are usually broken down into the combination of short segments (including single characters) after the segmentation process. Under such circumstances, an intuitive way to identify new words is to merge adjacent segments into longer n-grams [Tung 94, Wang 95], then use a separate filter for filtering out inappropriate candidates, and, at last, optionally applies a disambiguation step to remove candidates that are overlapped with other candidates in the sentences [Tung 94].

There are some problems with such a non-iterative segmentation-and-merging process which uses known words alone for finding unknown word candidates. First, there are two types of segmentation errors [Chiang 92, Lin 93a, 93b], namely over-segmentation (which segments unknown words into shorter segments) and over-merging (which combines short words into longer ones). The merging operation may not recover all mis-segmented words, because it can only recover unknown words that are over-segmented into small segments. Such an operation, however, cannot recover unknown words that are (fully or partially) embedded in known words. For instance, the following example in Section 1.4.2: '<u>土地</u> <u>公有</u> <u>政策</u> ...' ('The policy for the land to be owned by the public...') is likely to be segmented into '<u>土地公</u> <u>有</u> <u>政策</u>' ('The Local Village God has his own policy...') if we don't have the unknown word '公有' ('public ownership') in our vocabulary, and instead, we have a known word '土地公' ('the Local Village God') which includes the first character of the unknown word '公有'. In this case, merging the adjacent segments '土地公', '有' and '政策', will never produce the unknown word candidate '公有'. Second, a large number of merged segments will be produced, which are unlikely to be produced if a reasonably good augmented dictionary is used during the segmentation process. Submitting such a large candidate list (which amounts to about 2 million entries for our corpus) to the filters will degrade the precision of the system. Finally, an extra disambiguation process, as described in Section 1.4.1, might be necessary to resolve ambiguities between overlapping candidates.

To avoid the above problems, we can construct an *augmented dictionary* to include potential unknown words to the vocabulary in addition to known words. And make the known words and potential unknown words compete with each other under the contextual constraints imposed by the segmenter. It could then be expected that the number of words in the output list (derived from the segmented corpus) will be greatly reduced after applying the contextual constraints within the sentences, because only 'likely' unknown words (in the maximum likelihood sense) among all potential unknown words (in the augmented dictionary) will be submitted to the filter. With our corpus, the number of words derived at the initial iteration by using the augmented dictionary is only about 1/15 that of the merged word candidates (which are acquired by segmenting the corpus with known words only and merging adjacent n-grams afterward.)

To focus on the extraction of significant new words that occur frequently, the initial *augmented dictionary* used for segmentation consists of single characters in the corpus, the words in the system dictionary, and *n-grams* whose frequency of occurrence is at least 5 times in the text corpus. The augmentation thus is able to recover not only over segmented unknown words but also short words that are embedded in known words. Furthermore, only bigrams, trigrams and quadragrams are included in the augmented dictionary since most Chinese words are of length 2, 3, or 4 characters.

By applying MLE (maximum likelihood estimation) over the segmentation patterns with the augmented dictionary, the most likely n-grams (in ML sense) would include not only the system dictionary words but also the most likely unknown words. To view it in another way, using the augmented dictionary would, in some sense, merge (or split) known words and isolated single characters into potential unknown word candidates; and such candidates are more likely to be words than those millions of randomly merged n-grams, because they are selected based on the ML criterion under the contextual constraints embedded in the input corpus. Consequently, only such highly likely n-grams, instead of those millions of randomly merged n-grams, will be submitted to the filter for further justification; hence the precision of the system is expected to be raised significantly. Besides, it is also unnecessary to apply an extra disambiguation step for removing candidates that are overlapped with other candidates in the sentences ([Tung 94]), since the segmentation processes will automatically determine which n-grams to survive based on the ML criterion, and thus resolves such ambiguity implicitly.

4.6 Performance Evaluation

To show how effective the unsupervised Viterbi word segmentation module is, the performance of only using the segmentation module is listed in Table 20. It is observed that the training process converges very quickly using the above Viterbi training procedure. (See Figure 4 at the end of this section for the bigram example.) The performance for the initial iteration (iteration #1), the second iteration and the 13th iteration (where the system converges) is listed in the table.

n-gram	iteration number	p (%)	r (%)	WPR	FM
2	1	65.83	72.75	69.29	69.12
	2	68.67	76.67	72.67	72.45
	13*	68.72	78.41	73.57	73.25
3	1	26.45	78.64	52.55	39.59
	2	28.81	80.68	54.75	42.46
	13*	29.63	81.36	55.50	43.44
4	1	36.57	93.09	64.83	52.51
	2	38.24	93.45	65.85	54.27
	13*	38.96	93.09	66.03	54.93

Table 20 Performance of the word segmentation module using the Viterbi training procedure for identifying new words (*: performance at converge; convergence is reached at iteration #13.)

Table 20 shows that the precision rates after convergence for the extracted new words of 2, 3, and 4 characters long are about 69%, 30%, and 39%, respectively. And the recall rates for the new words are about 78%, 81%, and 93%, respectively. The joint performance in WPR are about 74%, 56% and 66%, and FM achieves the rates of 73%, 43% and 55%, respectively, for 2-, 3-, 4-character new words. This means that most (78-93%) new words are included in the extracted lists, and one can pick up a correct new word from the lists about every 1.5 - 3.4 entries. Such an unsupervised segmentation-only model is therefore a useful tool, by itself, for extracting new words from the corpus.

To give us some feeling whether such performance is good, it was compared with other previous works. Since most other works we had investigated either do not provide the precision and recall performance at all, or simply provide an estimate on precision rate without giving an estimate for recall rate (which is usually costly to estimate), we can therefore only quote the reported results in [Wang 95] for a very rough comparison. (The criteria for recognizing an n-gram as a lexical entry by different lexicographers could be different though. Therefore, the following comparison may not be a fair comparison, since they are not based on completely identical environments.) In [Wang 95], a measure similar to the *strength* statistic used in the Xtract system [Smadja 93] was used to extract new words. The corpus is of the same domain as ours (i.e., news articles) and has about the same size. The reported new word precision rates are about 22%, 28% and 12% for bigram, trigram and quadragram Chinese new words. (The new word recall rate is not reported in the above mentioned work. Note also that the terms ‘bigram’, ‘trigram’ and ‘quadragram’ in [Wang 95] are not completely identical to ‘2-character’, ‘3-character’ and 4-character’ strings, respectively. But the distributional analyses and examples given in [Wang 95], such as ‘trigram proper names’, ‘bi-collocation’ seems to suggest that, most of the time, they are equivalent. Even with such variation, it is expected that the precisions for bigrams, trigrams and quadragrams in the above work should not exceed 30%.) With the high recalls and much higher precision rates, the proposed Viterbi training approach should be competitive. We will address how to improve such performance even further in later sections. (In the improved scheme, it is possible to increase the precision rates to about 72% (bigram), 39% (trigram) and 56% (quadragram) after 21 iterations.)

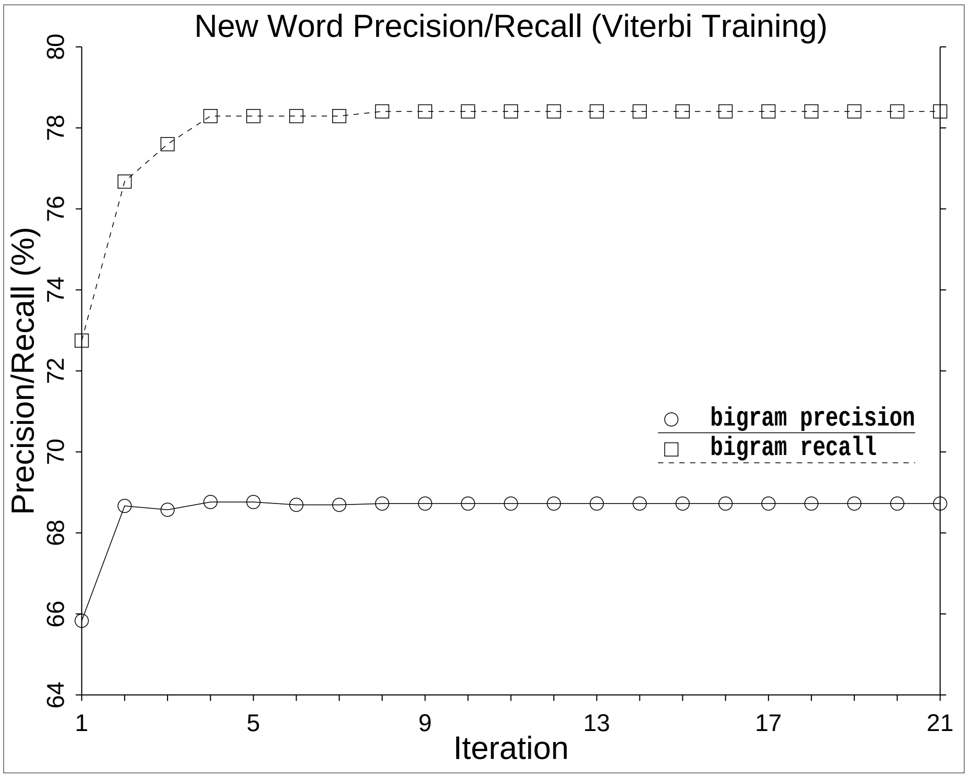

Figure 4 Performance of each iteration for bigram new words with Viterbi training model

To show how the Viterbi Training converges, the performance of the bigram new words for each iteration is shown in Figure 4. It shows that both the precision and recall of the bigrams are improved as the Viterbi training goes, instead of sacrificing precision for recall or *vice versa*, as is often observed in other simple filtering approaches. Note that, the largest changes in precision and recall occurs between the first and the second iterations. No further significant change is observed in the performance curves. This means that two iterations are usually enough for Viterbi Training. This situation is also shown in Table 20, in which the differences between the second iteration and the converged results is negligible.

4.7 Problems with Using Segmentation Module alone

In spite of the encouraging results obtained in the last section, the above model has one main problem: the likelihood function used in the maximization process consults only n-gram frequency information. No other character association features (and syntactic or semantic information) are consulted during the word segmentation process. Therefore, the 'most likely' segmentation thus acquired is simply the one which has the largest product of relative word frequencies among all segmentation patterns. However, the 'new words' extracted in this way may not have the desirable characteristics such as 'having high mutual information', 'containing no special characters (such as 'de' (的), 'zhi' (之)), symbols, and Chinese quantifiers', and so on.

In other words, the words extracted from the 'most likely' segmentation pattern based on the above likelihood function may not coincide with the lexicographers' general sense of *most likely words*. For instance, the bigram '的人' ('people who ...'), which occurs 962 times in the corpus, may be recognized by the word segmentation module as a word simply because it occurs frequently. As a result, many n-grams which will not be qualified by the lexicographers as words may be extracted in the word segmentation process simply because they occur frequently, and the precision rate will be low against the lexicographers' criteria. (For more examples and quantitative analyses, please refer to Section 5.6 "Error Analyses".) Therefore, it is desirable to use other information, such as some association measures, to remove such n-grams from the augmented dictionary.

4.8 Truncating Inappropriate Candidates with a Likelihood Ratio Ranking Model

To filter out unlikely candidates, the minimum error classifier as described in Chapter 2, is implemented in the current work as our 'filter' for supplementing the Viterbi word segmentation module. In other words, we use the *log-likelihood ratio* (LLR) as a *joint* association metric of the other commonly used association metrics, and use this joint metric to tell which n-grams are more likely to be words, and which are more unlikely to be words among all n-grams. In the current task, we have

$$\lambda = \frac{f(\mathbf{x}|\ \mathbf{W})P(\mathbf{W})}{f(\mathbf{x}|\ \overline{\mathbf{W}})P(\overline{\mathbf{W}})}, \quad \text{LLR} \equiv \log \lambda$$

where λ is the likelihood ratio for the n-gram, $\log \lambda$ is the log-likelihood ratio (LLR), $f(\mathbf{x}|\mathbf{W})$ (or

$f(\mathbf{x}|\overline{\mathbf{W}})$) is the density function of the feature vector $\mathbf{x}$ in the word-class (or non-word class), and $P(\mathbf{W})$ (or $P(\overline{\mathbf{W}})$) is the prior probability for the corresponding class $\mathbf{W}$ (or $\overline{\mathbf{W}}$). If $\lambda \geq 1$, i.e., the log-likelihood ratio LLR ≥ 0 (or larger than a threshold log λ_0) for an n-gram, then it is classified as a word; otherwise, it is classified as a non-word. This model is used as our basis for the *likelihood ratio ranking module* (LRRM, or 'ranking module' for short) in Figure 2.

To estimate the density functions and the prior probabilities, we must have two sets of training n-grams, one of which has all its class members assigned the word-class label, and the other set has class members that are labeled as non-word. However, since the n-grams in the input text corpus are not associated with their word/non-word class labels, the initial class labels of the feature vectors are obtained by dividing the n-grams of the input text corpus into the word and non-word classes, depending on whether an n-gram can be found in the system dictionary or not.

Many discriminative features, such as mutual information [Church 90, Wu 93, Su 94], entropy [Tung 94], dice metric [Smadja 96], relative strength [Smadja 93, Wang 95] and χ^2 testing statistics [Papoulis 90], can be used in estimating the LLR's. Since the mutual information and entropy features are found to be useful for English compound word extraction, they are used in the current task. (Note that we don't have part of speech information in the current task, and thus cannot use the POS discrimination feature as in the English compound word extraction task.) The definitions of the two association metrics are given in Section 2.5.2 and Section 2.5.4, respectively.

The joint density functions $f(\mathbf{x}|\mathbf{W})$ and $f(\mathbf{x}|\overline{\mathbf{W}})$ in the log-likelihood ratio are modelled as a mixture of multivariate Gaussian distribution (see Chapter 2), and the prior probabilities, mean vectors, and covariance matrices (i.e., the *parameters*) for the two classes are estimated using the mixture estimation method [Duda 73], as described in Chapter 2. By using such a density function, the features will be jointly considered for decision making, and the correlation between the two features are taken into account in terms of the covariance matrices. In the current work, 3 Gaussian mixtures (M=3) are used with 2 features (D=2) for the multivariate Gaussian density function of each class. Of course, if the density functions are not a mixture of multivariate Gaussian, other families of density function may have to be used as in the English compound extraction task. To focus our attention on the iterative precision-recall maximization scheme, such general parameter estimation issues will not be addressed further in this task.

4.9 Ranking-Module-Only Performance

The likelihood ratio ranking module (LRRM) can be used alone to identify n-grams as words or non-words. Therefore, it is interesting to see how this module performs by itself, before combining it with the segmentation module. One intuitive way to apply the ranking module is to use it in the unsupervised mode of operation as follows. Initially, the system dictionary is used to assign the word and non-word labels to all n-grams. The parameters for the two classes are then estimated according to such class labels. We then use $\log \lambda = 0$ as the threshold to identify the n-grams as word or non-word, and use such new class labels to re-estimate the parameter values. Table 21 shows the following estimated performance (against n-grams in the 1,000 sample sentences described in Section 4.2) after 21 iterations.

n-gram	p (%)	r (%)	WPR	FM
2	54.28	90.99	72.63	68.00
3	33.78	63.01	48.40	43.98
4	51.17	81.42	66.30	62.84

Table 21 Performance of the likelihood ratio ranking module (two-class classifier) (with $\log \lambda = 0$ as the threshold) for identifying unknown words

In comparison with the results in Table 20, where only the Viterbi training is applied for word segmentation, the bigram WPR using the ranking module alone is slightly worse and FM is worse by 5%. For trigrams, the slightly better precision is traded off by the large loss in recall; therefore, the WPR is smaller by 7% and only 0.5% is gained in FM. For quadragrams, the precision is higher (by 12%) but the recall is worse (by 12%), which results in a slightly larger WPR but a larger difference (8%) in FM. It is hard to conclude whether using the ranking module alone is better than using only the word segmentation module from such differences, especially when we are not sure whether the threshold is the best for the various performance indices; it depends on whether you are comparing the WPR or FM and whether you are comparing bigrams, trigrams or quadragrams (although the two models both are better than a few other reported works.) However, the following reasons suggest that using the ranking module alone is not a good idea under the current unsupervised mode of training.

First, the parameters of the ranking module may not be reliably estimated without having a large training corpus with correctly assigned word class labels. Second, the ranking module by itself does not take advantage of the contextual constraints for identifying new words; in other words, each n-gram is identified independently of its neighboring n-grams. Therefore, some mis-classified instances, such as the '委會同' example in Section 1.4.2, could be avoided if the contextual constraints are taken into consideration. Finally, the LLR=0 threshold may not be the best threshold for achieving best precision, recall, WPR or FM (although it normally provides a good starting point, in minimum error sense, for training the system parameters toward best precision/recall performance, as described in Chapter 2 for the English compound word extraction task.) Therefore, the ranking module will not be used as a stand alone classifier in the current work. Instead, it will be used simply as a ranking device, as described in Chapter 1, to supplement the segmentation module for acquiring better segmentation patterns and better word lists.

4.10 Combining Word Segmentation Module and Ranking Module

One way to make the ranking module enjoy the contextual constraints provided by the word segmentation module is to cascade the word segmentation module and the ranking module together, and use the ranking module to truncate unlikely new words from the output of the segmentation results (not from the augmented dictionary as we will do in the proposed method of Chapter 5.) Table 22 shows the performance by applying the Viterbi Training process twice to get the segmentation output and then truncating the worst 10% (in terms of their LLR ranks) of the new words extracted from the segmentation output. The Viterbi training is applied only twice since further iterations do not significantly change the segmentation performance (as mentioned in Section 4.6). Furthermore, instead of using an absolute threshold for the log-likelihood ratio test, we are using a *relative mode* of filtering to truncate only the most unlikely 10% new words from the output of the segmentation, since the best thresholds (in terms of maximum WPR or FM) cannot be obtained in the current status. To see the effects, the performance of the segmentation-only (WS-only) model (Table 20) and the ranking-module-only (LRRM-only) model (Table 21) are re-cited here for comparison. The performance of this cascading scheme is entitled ‘Non-Iterative’ in the table, since each of the word segmentation module and the ranking module is applied only once. The shaded cells in Table 22 designate the best WPR or F-measure performance of the various models.

n-gram	Models	p (%)	r (%)	WPR	FM
2	WS-only	68.72	78.41	73.57	73.25
	LRRM-only	54.28	90.99	72.63	68.00
	Non-Iterative	73.56	73.90	73.73	73.73
3	WS-only	29.63	81.36	55.50	43.44
	LRRM-only	33.78	63.01	48.40	43.98
	Non-Iterative	31.90	80.34	56.12	45.66
4	WS-only	38.96	93.09	66.03	54.93
	LRRM-only	51.17	81.42	66.30	62.84
	Non-Iterative	42.38	93.09	67.74	58.25

Table 22 Performance by cascading the segmentation module and the ranking module, in comparison with other models (WS-only: segmentation only; LRRM-only: likelihood ratio ranking module only; Non-Iterative: cascading the two modules and truncating the most unlikely 10% words from the segmentation output). (Note: WS model applies 13 iterations to converge, and LRRM model is iterated by 21 iterations.)

From the above table, the performance by cascading the two modules outperforms the segmentation- only model (which converges at the 13th iteration) and the ranking-module-only model (which iterates 21 times) in terms of WPR and FM, with only one exception. There are some implications from the above observation. First, the cascading scheme does have some degree of integration between the segmentation module and the ranking module, as can be expected in most such cascading scheme. In comparison with Table 20, it is easy to see that the improvement is gained by sacrificing a little recall for higher precision. Second, we can truncate unlikely candidates using the *relative mode of filtering* without really depending on an optimal threshold (in whatever sense) under the current unsupervised mode of training. For this reason, the classifier actually operates as a likelihood ratio ranking module in our system, which works in the relative mode of filtering. Note that, the recall rates, which might drop slightly in the filtering step will be compensated by using an iterative scheme, as described in the following chapter, so that both precision and recall can be improved at the same time.

Chapter 5
An Iterative Approach for Integrating the Word Segmentation Module and the Likelihood Ratio Ranking Module

5.1 Problems with Non-iterative Approaches

Although the cascading scheme in the previous chapter for combining the word segmentation module and the filter (ranking module) is easy to implement, it is non-iterative in the sense that there is no feedback path to the previous stage. As a result, the information provided by the filter cannot be used to enhance the power of the segmentation module. On the other hand, the parameters of the filter are estimated independently of the segmentation results; hence, the segmentation output has no contribution to the performance of the filter. The information provided by the two modules thus cannot fully supplement and enhance each other.

To address this problem, note that the performance of the word segmentation module greatly depends on how good the potential unknown words are included to the augmented dictionary; and the performance of the filter greatly depends on how good the parameters for the filter are estimated. Ideally, we should use a dictionary consisting only of the words embedded in the input corpus to get the desired segmentation. However, it is impossible to have such a dictionary in advance. Therefore, the initial augmented dictionary is constructed simply by combining the system dictionary and n-grams that occurs at least 5 times in the input corpus. In other words, the association information on each entry of the augmented dictionary is not consulted in constructing the augmented dictionary. On the other hand, if the true class labels of the n-grams in the input corpus are known in advance, then the estimated filter parameters will be close to the desired values. However, we can only rely on the system dictionary to assign word-class labels with certain; the other n-grams not in the system dictionary may be incorrectly labelled. As a result, the parameters of the filter is estimated independently of the segmentation results.

Therefore, it is desirable to use the filter to improve the performance of the segmentation module by refining the augmented dictionary. It is also desirable to refine the class labels of the n-grams in the input corpus (and thus the estimated parameters of the filter) by using the new word list suggested by the segmentation module. Unfortunately, in the non-iterative cascading scheme described in the last chapter, the augmented dictionary used for segmentation is not refined by the filter, and thus cannot take advantage of the association information provided by the filter to improve the segmentation results. On the other hand, the word and non-word labels assigned to the n-grams for estimating the parameters of the ranking module are based solely on the system dictionary. Such word-class n-grams are only about 10% of all the n-grams; and the remaining 90% of n-grams, including unknown words, are all labelled as non-words. Therefore, the initial class labels are biased and they would introduce many mistakes; hence, such labels will prevent the system from estimating a good parameter set to achieve good results.

To improve the system performance, it is, therefore, desirable to provide a better augmented dictionary to the segmenter, and estimate better model parameters for the filter. Since a refined augmented dictionary can be constructed with the aid of the filter, and a better parameter set can be estimated by taking advantages of the improved segmentation results, it is highly desirable to use the output of each module to iteratively enhance the performance of other modules, until no further improvement is observed. Therefore, it is possible to improve the system performance if there is a way for forming a feedback path to enhance each other. The details will be addressed in the following sections.

5.2 The Iterative Integration Method

To improve the one-pass non-iterative scheme described earlier, the following iterative approach is proposed to integrate the word segmentation module and the likelihood ratio ranking module (LRRM). In each iteration, the word segmentation result is improved by using a refined augmented dictionary, which was produced by truncating the most unlikely word candidates in the augmented dictionary of the last iteration with the aid of the ranking module. Such improvement by refining the augmented dictionary is possible, since statistical word segmentation can achieve very satisfactory results (over 99% word segmentation accuracy [Chiang 92, Lin 93b]) if all unknown words are included in the system dictionary and no spurious words are included in the system dictionary. On the other hand, the parameters of the ranking module are improved at the end of each iteration by updating the word and non-word class labels of the n-grams; this is accomplished by re-labeling the class labels of a small fraction of the most likely words identified by the word segmentation module. The performance of the ranking module can thus be improved by using progressively improved word/non-word class labels derived from the word segmentation output using the contextual information of the input corpus.

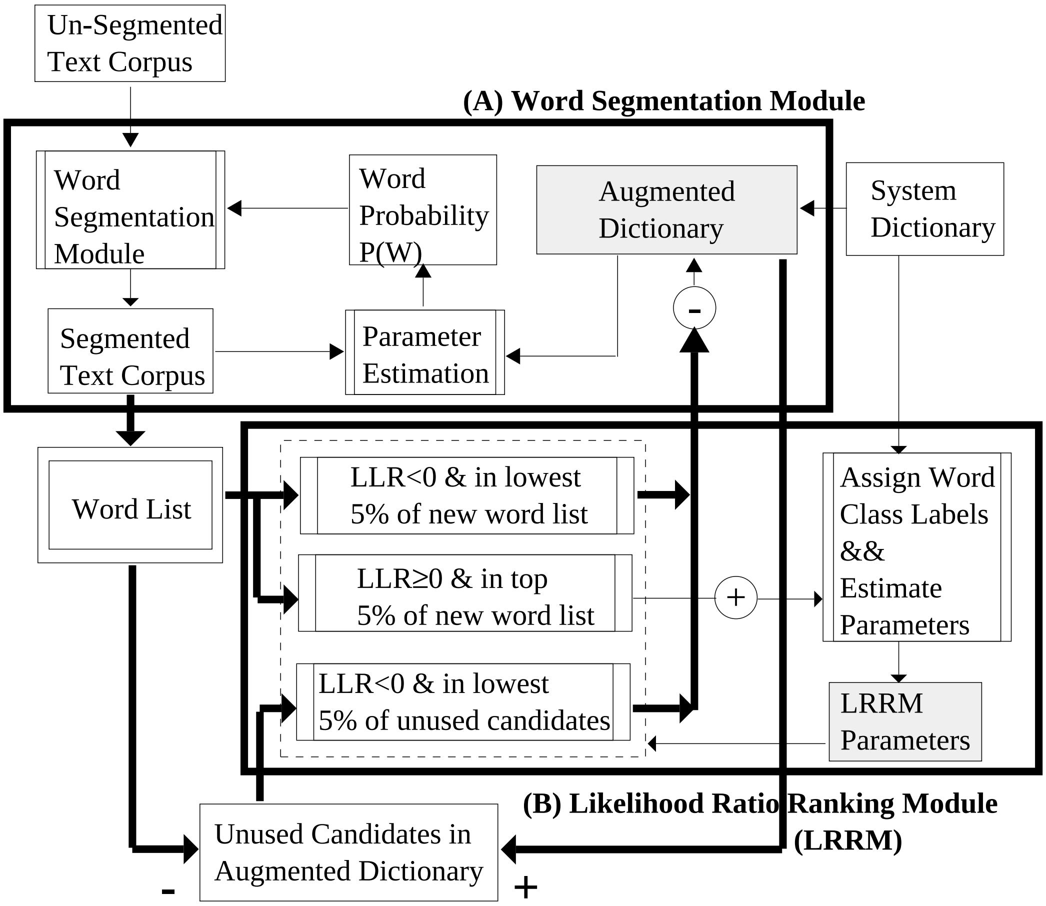

Figure 5 Iteratively integrating word segmentation module and likelihood ratio ranking module.

Figure 5 illustrates the iteratively integrated system for unknown word identification. Box (A) is the segmentation module as described in the previous chapter. Box (B) corresponding to the likelihood ratio ranking module, which is used to estimate the log-likelihood ratios for various n-grams in order to update the augmented dictionary, and to update the word/non-word labels of the n-grams in the input corpus for re-estimating the parameters of the ranking module. The three processes in the dashed boxes within the ranking module represents three updating actions (to be described later) for updating the augmented dictionary of the segmentation module and the parameters of the ranking module.

As described earlier, the initial augmented dictionary is constructed by combining the system dictionary and n-grams in the input corpus that appears more than 5 times (inclusive); their initial word probabilities are estimated as their relative frequencies in the input corpus. For each such n-gram, it is initially labeled as ‘word’ if it belongs to the system dictionary; otherwise it is labeled as ‘non-word’. The initial parameters of the filter is then estimated based on the initial class labels. Once the initialization is complete, the input corpus is segmented, and its output word list is used for updating the augmented dictionary and filter parameters; afterward, the refined augmented dictionary and filter parameters are used for re-segmentation and re-ranking, iteratively.

Note that, the Viterbi training process for word segmentation converges very quickly, and the largest change in performance occurs between the first iteration and the second iteration; further iterations do not significantly change the best segmentation results. (See Table 20 for details.) We therefore apply the Viterbi training only twice in the word segmentation process to speed up the processing time.

The ranking module takes an n-gram list as its input, estimates the log-likelihood ratio (LLR) of each n-gram in the list. The log-likelihood ratios are then used to rank the n-grams and indicate which one is ‘more likely’ or ‘more unlikely’ to be a word; a small fraction of the input list is then used for rejection or acceptance. More precisely, 3 sublists (corresponding to the three updating actions in Figure 5) will be submitted to the ranking module for ranking (by their LLR’s); 2 such sublists then will be used to update the augmented dictionary and one sublist will be used to update the parameters of the ranking module as follows.

(1) The new word n-grams derived from the segmented corpus whose LLR is less than zero

(LLR<0) are sorted by their LLR's; the worst 5% (in terms of LLR) of this sublist are truncated from the augmented dictionary to refine the augmented dictionary. This step is intended to remove the most unlikely candidates in the new word list from the augmented dictionary using the global character association metrics. Such candidates are preferred by the segmentation module but unqualified by the ranking module; therefore, they should be removed to prevent them from appearing in the word segmentation output again in later iterations, so that the segmentation results in later sections will be improved.

For instance, the non-word substring '不足爲' is identified as a new word in the incorrectly segmented phrase '不足爲 大哥' ('is not qualified as the Big Brother'), since it also appears frequently in words like '不足爲奇', '不足爲由', '不足爲信', '不足爲惜' or '不足爲慮' ('is not really strange', 'cannot be taken as the appropriate reasons', 'cannot be taken as the evidence', 'is not pitiful' or 'is not worthy to worry about')). However, the non-word string '不足爲' (with LLR= -7.73) will not be qualified by the ranking module; therefore, it will be removed by this updating action, and the better segmentation '不足 爲 大哥' is acquired after this non-word substring is removed. (Also recall the '(移送)台中少年法庭審理' example in the Introduction section, which acquires the correct segmentation '(移送) 台中 少年法庭 審理' after two successive removals of the non-word strings '台中少年' and '法庭審理'.)

(2) The potential unknown words (in the augmented dictionary) which do not appear in the segmented corpus form a sublist of 'unused candidates'; among such 'unused candidates' whose LLR<0, the worst 5% n-grams (in terms of LLR) are also excluded from the current augmented dictionary; such entries are removed because they are neither preferred by the word segmentation module nor preferred by the classification modules. Hence, they are very unlikely to be new words. (The '+' and '- ' signs near the box entitled 'unused candidates in augmented dictionary' means to form the 'unused candidates' by including ('+') potential unknown words in the augmented dictionary and then excluding ('-') entries that appears in the output word list of the segmentation module.)

Most such n-grams are random n-gram strings such as ‘元遠期’ (which is a substring of ‘美元遠期外匯’ ‘US dollar long-term exchange rate’). Removing such entries will reduce the computation cost without introducing significant errors. Given the refined augmented dictionary updated according to the above two sublists, the Viterbi training will be applied again, in the next segmentation-filtering iteration, to segment the corpus with the refined augmented dictionary.

(3) A third sublist is used to facilitate the estimation of the filter parameters as follows. The parameters of the classifier are trained by dividing n-grams in the input corpus into the word and non-word classes and estimating the parameters from the distributions of these two classes. Initially, only those n-grams in the system dictionary are assigned the word-class label for training the word-class parameters, and the others are assigned the non-word label for training non-word-class parameters. To improve the parameter estimation of the likelihood ratio ranking module, those n-grams which had been assigned the word-class labels in previous iterations are first excluded from the new word list (as described in (1)); a small fraction of the remaining new words, which have the highest LLR’s among all, are then re-assigned to the word-class so that the parameters of the ranking module are re-estimated based on the new class labels. The number of re-assigned new word candidates is 5% the size of the n-grams which had already been assigned the word class labels. In other words, the number of n-grams which are assigned the word-class label will be increased by 5% per iteration. The reason for setting this fraction according to the current word-class size is to ensure that the number of new word-class members will not grow too fast as to overwhelm the number of original word-class members (which contains only the system dictionary entries) too much.

Since such n-grams are preferred by both the segmentation module and the ranking module as words, they are combined with the original system dictionary entries for training the word-class parameters. The remaining entries are used for training non-word-class parameters. After the new word-class labels are re-assigned, the parameters of the ranking module are re-estimated so that they can be used for evaluating the LLR’s of various n-grams in the next iteration.

Since the initial class labels assigned to the n-grams for training the ranking module is solely based on the system dictionary, the initial LLR's may not be estimated reliably to reflect the true ranks of the n-grams, especially for n-grams which neither have very high LLR's or very low LLR's. Therefore, it would be safer to use only a small percentage for updating the system, so that the real new words are less likely to be truncated and the recall can be kept high; and spurious words are less likely to be assigned the 'word-class' label to keep precision high. Currently, we are truncating those very unlikely candidates among the worst 5% candidates determined by the ranking module, and increasing the size of the word-class members by 5% per iteration. The effects of adjusting such percentages will be surveyed in our future works. Since the fractions are small, the corresponding sublists will consist almost entirely of words or entirely of non-words. As a result, the system will be refined iteratively without introducing considerable errors.

5.3 Performance Evaluation on the Integrated System

Table 23 shows the estimated performance of the word segmentation module of the iteratively integrated system based on the new word lists extracted from the 1,000 sample sentences. The performance at iteration 1 is the performance before the augmented dictionary is iteratively refined; in other word, it's the performance of the segmentation module of the non-iterative scheme. Table 23 shows that the recalls for the embedded *new words* are as high as 81%, 88% and 94%, respectively for 2-, 3-, 4-character words after 21 iterations. This means that more than 81% new words will be extracted from the text corpus. Only a small portion of them (6-19%) will be lost. This table also shows that the output new word lists have precision rates of 72%, 39%, and 56%, which are reasonably high for lexicography applications. The joint performance for new word identification, in terms of WPR is 77%, 63% and 75%. The F-measures are 76%, 54% and 70% respectively. With such performance, it is believed that the proposed integration method could provide a good tool for lexicographers to identify new words with considerable saving in human efforts.

n-gram	iteration number	p (%)	r (%)	WPR	FM
bigram	1*	68.67	76.67	72.67	72.45
	21	72.39	80.83	76.61	76.38
	Difference	3.72	4.16	3.94	3.93
trigram	1*	28.81	80.68	54.75	42.46
	21	38.60	87.80	63.20	53.62
	Difference	9.79	7.12	8.45	11.16
quadra-gram	1*	38.24	93.45	65.85	54.28
	21	56.21	93.82	75.01	70.30
	Difference	17.97	0.37	9.17	16.03

Table 23 Performance of the word segmentation module in the iteratively integrated system.
(‘Difference’: difference in performance between iteration 21 and iteration 1.)
(*Note: the performance of iteration 1 corresponds to the output of the word segmentation module in the non-iterative scheme.)

The effect for iteratively updating the augmented dictionary is shown in the rows entitled ‘Difference’ in Table 23, which are the differences in performance between the 21st iteration and the first iteration (i.e., non-iterative performance). Note that, *both* precision and recall (for all n-grams) are improved at the same time, as expected. The improvement by incrementally truncating the vocabulary of the initial augmented dictionary and re-maximization in the integrated system is significant, especially for 3-character and 4-character words. As can be seen in the above tables, the precision is improved by about 4% (bigram), 10% (trigram), 18% (quadragram). The recall is improved by about 4% (bigram), 7% (trigram), and 0.4%(quadragram). (The improvement in recall for quadragram new words is small since more than 93% of them have been identified and the number of quadragram new words is smaller than that of bigram or trigram new words.) The improvement in WPR is 4%, 8%, 9% for bigram, trigram and quadragram, respectively; and the F-measure is improved by 4% (bigram), 11% (trigram), and 16% (quadragram), respectively.

To show whether the smaller improvement for the bigrams are statistically significant [Wonnacott 90], a hypothesis testing is conducted to see whether the difference before and after applying the iterative refinement processes really make difference. (See the appendix for the hypothesis testing being conducted.) According to the testing statistic, we can conclude that, with more than 99.8% confidence, that the smaller improvement for bigrams is due to the adoption of the iterative scheme, not due to sampling variation in estimating the performance. (See also the error analysis sections for further quantitative analyses.)

Figure 6 shows the change in precision and recall for bigram words in each iteration. The performance increases almost monotonically while progressively removing low LLR candidates from the vocabulary. Therefore, the proposed method provides a way to stably improve the precision and recall without trading one performance for the other.

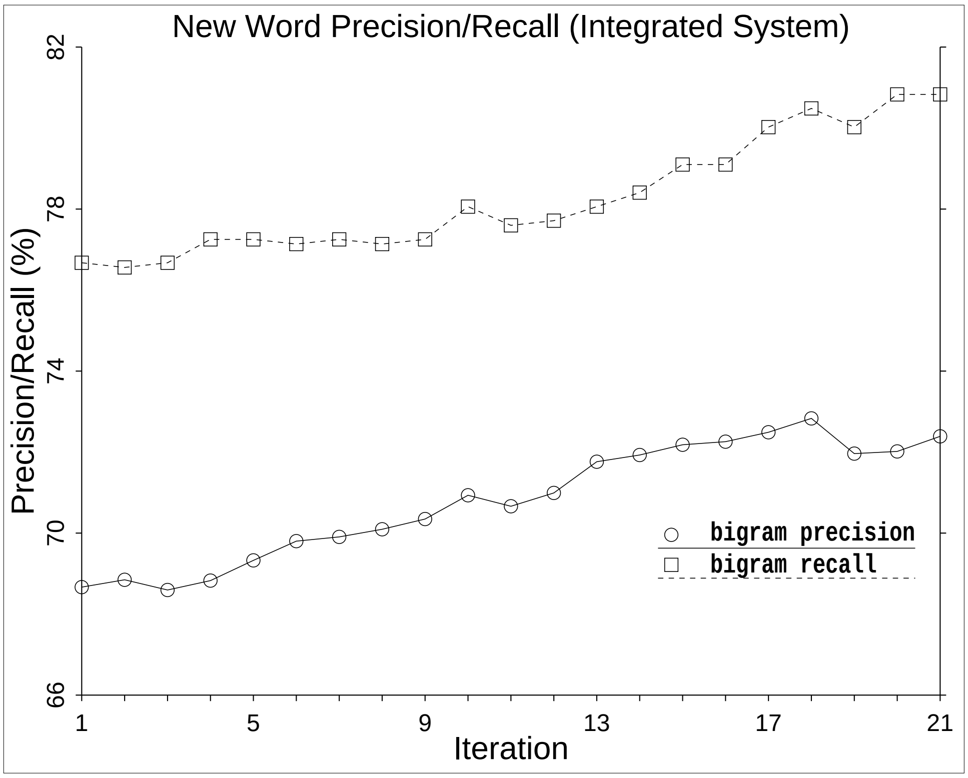

Figure 6 Performance of each iteration for bigram new words in the integrated system.

As for the computation load, each iteration of the integrated system takes about 6 hours of processing time using 2 Sun Sparc10 workstations and one IBM RS/6000 workstation concurrently (through a heavily loaded network with large amount of I/O operations). Such processing time is about the same order as in [Tung 94] (by scaling up the corpus size to the same amount). According to the *precision* rates in Table 23 and the *sizes* of the output word lists of the segmentation module at the final iteration, it is estimated that we can extract about 12,600 bigram new words, 8,400 trigram new words and 8,500 quadragram new words, i.e., 29,500 new words in total, from the 311,591 sentences, including compound words and proper names. Considering the costs for manually identifying the large corpus, such computer processing time is quite affordable. Furthermore, since the estimated new word recalls do not have significant change near the final iteration and only the precision of the trigrams have a more visible change near the final iteration, the process is thus terminated at the 21st iteration without waiting for true convergence. Hence, some of the above conclusions may not be extrapolated to later iterations. The percentages for truncating unlikely n-grams from the augmented dictionary can be adjusted to affect the convergence speed. However, the effect of varying this factor is beyond the scope of this work, and will not be addressed here.

5.4 Quantitative Analyses for the Performance Improvement

There are several factors affecting the precision and recall rates of the system. The following analysis will further justify our strategy to improve the system performance by iteratively truncating unlikely candidates from the augmented dictionary. Let n_{ww} be the number of true new words that are in the output new word list, n_{wx} be the number of true new words that do NOT appear in the output new word list; and let n_{xw} be the number of non-word n-grams that are in the output new word list. (The first subscript in the symbols indicates whether an n-gram is a real new word or not, and the second subscript indicates whether the n-gram is identified by the system as a new word or not, where 'w' means 'word' and 'x' means 'non-word'.) The new word precision and recall as defined previously is then equivalent to:

$$p = \frac{n_{ww}}{n_{ww} + n_{xw}} = \frac{1}{1 + n_{xw}/n_{ww}}$$

$$r = \frac{n_{ww}}{n_{ww} + n_{wx}} = \frac{1}{1 + n_{wx}/n_{ww}}.$$

Table 23 shows that most contribution of the F-measure and WPR comes from the improvement in precision, especially for trigrams and quadragrams. Such improvement could come from an increasing n_{ww} or a decreasing n_{xw}. Further detailed investigation shows that n_{ww} (i.e., the number of correctly identified new words) are *increased* by 5% for bigram, and by 8% for trigram between the initial and final iterations; and n_{ww} is almost constant through all iterations for quadragrams in the sample sentences (since 93% of the quadragrams had already been identified in the first iteration). On the other hand, n_{xw} (namely, the number of non-words that are incorrectly identified as words) are *decreased* by 12%, 30%, and 52% for bigrams, trigrams and quadragrams, respectively.

Since the decrease in n_{xw} is much larger than the increase in n_{ww}, the improvement in precision is mostly attributed to the decrease in n_{xw}. In other words, truncating unlikely candidates from the augmented dictionary is the major factor in improving the performance of the system. Such behavior well justify our previous arguments to improve the performance by truncating unlikely candidates from the augmented dictionary.

Furthermore, the truncated n-grams will be replaced with other likely shorter segments (such as changing '將前往' into '將 前往'), or make other combination of likely segments to appear in the best segmentation (such as changing '凱悅大 飯店' into '凱悅 大飯店', where the deletion of '凱悅大' make the better segmentation '凱悅 大飯店' to emerge). In more complicated cases, successive removals of unlikely n-grams will be conducted: such as changing '移送 台中少年 法庭審理' into '移送 台中 少年 法庭審理' and then into '移送 台中 少年法庭 審理', as given in the Introduction chapter. Therefore, n_{ww} will also be increased (and thus improve the precision further) due to the use of association metric (which reject unlikely candidates that were preferred by the word segmentation module.) Besides, because $n_{ww}+n_{wx}$ is a constant (corresponding to the number of real new words in the sample sentences), the increase in n_{ww} also correspond to the decrease (by the same amount) in n_{wx}, namely, the number of new words that are incorrectly identified as non-word. As a result, the recall is also improved due to the increasing n_{ww} and the decreasing n_{wx}. Thus, the precision and recall rates are increased almost monotonically.

5.5 Distribution of Unknown Words

The following table shows some randomly sampled new words in the output list of the 1,000 sample sentences. They are classified roughly according to [Wang 95] (which will be detailed in later quantitative analyses.)

Bigram New Words		**Trigram New Words**		**Quadgram New Words**	
Proper Names					
鹿谷	Lu-Gu; a county name	中新社	China News Service	曾蔡美佐	a female name
蓋茲	(Bill) Gates	富士通	Fujitsu	新興分局	Hsin-Hsing police office
住友	a company name	翁秀卿	a female name	富岡國小	Fu-Gang Primary School
Ordinary Words					
護法	guard	管理局	Bureau of Administration	年度預算	annual budget
幹員	talented (police) men	養豬戶	pig-raising farmers	全球股市	global stock markets
鑑於	in view of	下半年	second half of the year	貨幣市場	monetary market
共舞	dance (with somebody)	投機風	opportunism	國家公園	national park
責令	command	收盤價	closing price	生命安全	personal security
Abbreviation					
市警	city policemen	國台辦	Taiwan-Affair Office of National Affair House	省都委會	provincial city development committee
中菲	Sino-Philippine	消基會	the Consumer Protection Committee	紅會人員	the Red Cross staffs
鄉代	county representatives	上下班	go-to-and/or-come-back-from the office	投開票所	polls
Collocational Strings					
就會	will then ...	據指出	it was indicated that ...	絕大多數	overwhelming majority
既非	neither	並沒有	do not	一片混亂	a mess
Derivational Words					
廠方	authority of the company	壽險業	life insurance companies	所有權人	owner
		複雜化	complicate		
Numerical Strings					
一萬	ten thousands	十四日	14th day of the month	八十年度	1991 accounting year

Table 24 Examples of the acquired new words in the 1,000 sample sentences at the final iteration.

To provide quantitative indication on what kinds of new words are extracted and what kinds of new words are incorrectly identified, the new word list of the 1,000 sample sentences at the final iteration are sampled and classified according to [Wang 95] into the following categories: proper names (P), abbreviational words (A), derived (derivational) words (D), collocational strings (C), numerical strings (#) and other newly coined words (O) (i.e., ordinary new words not generated through the special word formation mechanisms of the P, A, D, C, # categories). 100 samples are drawn from each n-gram new word list derived from the 1,000 sample sentences for estimating their distribution (except two lists which have fewer than 100 tokens in the 1,000 sentences).

The P, A, D, C type words represents new words that are generated from a few productive new word formation processes used by native Chinese people [Wang 95]. Other new words, which are not produced by such special mechanisms are categorized as the O (ordinary or others) type. In [Wang 95], P, A, D, C as well as ‘newly coined words’, and ‘ambiguous words’ are used for classification, however, the last two categories cannot be easily distinguished, we therefore classify them and other unclassifiable new words into the O category.

The collocational strings are character strings that are frequently used at the same time in Chinese text (such as ‘即告’ (in the context of ‘即告失敗’) , ‘就會’). Some of them may be further divided into shorter words. However, they are so frequently used that the Chinese lexicographers may want to include such lexical units into the dictionary ([Wang 95]). One reason for that is that their meanings may not be easily acquired through certain lexical rules or morphological rules. Such entries may also be arguable for different lexicographers, since the qualification heavily depends on the lexicographer’s criteria. For instance, the collocational strings as listed in Table 24 (and the examples such as ‘乃於’ and ‘一定是’ given in [Wang 95]) may not be unanimously approved by all lexicographers. However, the percentages of such debatable strings that are recognized as words are small, as given in Table 25, especially for trigrams and quadragrams in the current work. Furthermore, various approaches are tested against the same benchmark prepared by the same person, as described in Section 4.2. Therefore, the differences in performance among various approaches are very likely to reflect the true situation.

The numerical strings are also classified into one special category for quantitative analyses, because we don’t have any special treatment on the numerical strings in the current work, and thus could make mistakes on such strings. As we shall see in later quantitative analyses, numerical strings also contribute a large portion of the errors since they rarely follow the contextual constraints or the association metrics as used in the current work. However, the syntax for most numerical strings is regular; there is no doubt that such strings can be parsed out using a finite state machine to improve the reported performance further. Therefore we will not make any comments on such a category.

n-gram	P(%)	A(%)	D(%)	C(%)	O(%)	#(%)	Total (%)
2	9	2	0	16	67	6	100
3	34	5	21	7	23	10	100
4	5	4	1	5	82	3	100

Table 25 Distribution of correctly identified words (P: proper names, A: abbreviational words, D: derived words, C: collocational strings, O: other ordinary new words [Wang 95].)

Table 25 shows the distribution of various kinds of new words that are correctly identified. The gray cells in the table identify special categories that have larger percentages. The bigram collocational strings (C), 3-character proper names (P) and 3-character derived words (D) are particularly noticeable, except for the large portions of ordinary new words (i.e., O-type new words). The high percentage for 3-character proper names is due to the fact that most Chinese proper names are of 3-character long. Therefore, the current approach provides a good tool for extracting proper names. As for quadragrams, the largest portion is attributed to compound words which consist of two double-character words. The bigram O-type new words, on the other hand, are hard to be classified according to special word formation rules, since bigram Chinese words are the largest population in Chinese words, and it appears that they belong to many different syntactic variants.

5.6 Error Analyses

There are two types of errors in extracting the new words. The first type is to miss real new words (referred to as Type I error), and the second type is to recognize non-words as new words (referred to as Type II error). In the 1,000 sample sentences, 1099 errors are made; the Type I error resulted in 219 missed n-grams (76% of which are bigrams, 16% are trigrams and 8% are trigrams); the Type II error resulted in 880 mis-recognized non-word n-grams (including 30% bigrams, 47%

trigrams, and 23% quadragrams). Therefore, about 80% of the errors are attributed to the Type II errors. This means that we can still truncate more spurious words from the augmented dictionary to achieve better performance if we conduct more iterations. Furthermore, since most Type I errors are attributed to bigrams, we can clearly see, from Table 23, that trigram and quadragram words are recognized with very high recall (88% and 94%). A few errors of the two types are listed as follows for demonstration:

Type I Errors: 拖吊 (towing), 除權 (split of stock shares), 營收 (income), 分案 (filing another case), 天價 (the historical price), 工寮 (worker's houses), 代金 (substitute of money), 平均線 (moving average), 任立渝 (a weather reporter's name), 副都心 (alternative city center), 雙黃線 (double yellow lines (for delimiting car lanes)), 中央銀行 (the Central Bank), 工業局長 (Chief Officer of the Industry Bureau), 無償配股 (unpaid stock shares), 新聞媒體 (news media)

Type II Errors: 將是 (will be), 已將 (have been), 爲此 (for this purpose/reason), (高)黃秀 (a female name), (黃)文忠 (a male name), 應朝 (should ... toward), 並向 (and ... from), (漲少)跌多 ((rarely be appreciated but) frequently be depreciated), 之二是 (two of the ... are), 與台灣 (... with Taiwan), 目前正 (currently is doing something ...), 其二哥 (his secondary elder brother), 的現象 (the phenomena of ...), 的拖吊 (... towing), 各自的 (individual), 除權後 (after splitting), 額滿後 (after fully registered), 作業中 (during operation), 狀況下 (under ... situation), 將下降 (will decrease), 國際標準(組織) (ISO), 上週大盤 (the closing stock indices of the last week), 無法接受 (cannot accept), (鄉)代會主席 (chair of the committee of the county citizen representatives), 住基隆市 (live in Keelung City), 教育局說 (the officers of the Bureau of Education said).

(Note: the Chinese characters within the parentheses are those strings which must be patched with the un-parenthesized characters in order to form a non-spurious phrase.)

The distribution of such errors are outlined in Table 26, which shows the estimated percentages of missing new words. (Note that the numbers of such trigrams (36 entries) and quadragrams (17 entries) are both less than 100 samples; therefore, the estimated percentages for trigrams and quadragrams are not as reliable as the other estimates.) Among the unrecognized trigram and quadragram new words, most of them are proper names (or numerical strings); therefore, by improving the proper name recognition a little further, the proposed method is likely to achieve even higher recalls in trigram and quadragram words. (Table 27, to be examined soon, further justify this statement, since Type II errors for trigrams and quadragrams that are originated from proper names are not the major source of the Type II errors.) The Type I errors for bigrams are evenly distributed among the P (proper name), C (collocational string), and O (other ordinary new word) types. Examining a few instances of such un-recognized bigrams shows that some such bigrams are embedded in spurious trigrams having extra special characters (e.g., '(的)拖吊' and '除權(後)') or quadragram collocational strings which consist of two double-character words (like '(七月)營收'). This may suggest that such spurious trigrams and quadragrams (as will be examined in Table 27) should further be truncated to bring those un-recognized bigrams back to the best segmentation.

n-gram	P(%)	A(%)	D(%)	C(%)	O(%)	#(%)	Total (%)
2	13	6	4	26	37	14	100
3	25	8	5	3	17	42	100
4	24	5	0	0	24	47	100

Table 26 Distribution of missing new words (Type I Error) (P: proper names, A: abbreviational words, D: derived words, C: collocational strings, O: other ordinary new words, #: numerical strings)

Table 27 further shows the distribution for Type II errors, in which non-word n-grams (i.e., spurious words) are classified as words. The column titles P, A, D, C, O and # now refer to the categories of the major substring of such non-word n-grams. For instance, the P column shows the percentage of mis-recognized non-word strings which are mainly originated from proper names.

n-gram	P(%)	A(%)	D(%)	C(%)	O(%)	#(%)	Total (%)
2	25	0	0	47	12	16	100
3	5	0	0	13	59	23	100
4	10	3	2	41	20	24	100

Table 27 Distribution of spurious words that are recognized as words (Type II Error) (P: originated from Proper names, A: originated from abbreviational words, D: originated from derived words, C: originated from collocational strings, O: originated from other ordinary words, #: originated from numerical strings.)

The largest portion of the bigram spurious words comes from collocational strings whose individual characters often appear at the same time. Most (44 out of 47) such collocational strings need be further divided into two single-character words to fully decompose them into the smallest meaningful units (e.g., ‘將是’, ‘已將’, ‘爲此’). However, since they are so frequently used at the same time, even native Chinese may be confused whether such strings should be considered as one lexical unit. (As described earlier, some such collocational strings were indeed considered as words or lexical units, and were included in the statistics of Table 23.) Therefore, it depends more or less on the lexicographer’s sense on how useful to include such strings in the lexicon. The second largest source for bigram spurious words comes from the substrings of the 3-character proper names (e.g., (高)黃秀 (a female name), (黃)文忠 (a male name)). The remaining bigram spurious words are ordinary words derived by over splitting longer words. For instance, ‘跌多’ (‘frequently depreciated’) is acquired by splitting the preferred idiomatic expression ‘漲少跌多’ (‘be rarely appreciated and frequently depreciated’).

For trigrams, most spurious words are originated from ordinary words. Further investigation shows that most such trigrams are composed of an ordinary words and a special character. For instance, 27 n-grams out of the 59 O-type instances contain the ‘的 (de)’ character at the first or the last character of the trigrams (e.g., ‘的拖吊’, ‘各自的’). Other special characters include time and location markers, such as ‘後’ (‘after...’, e.g., ‘除權後’, ‘額滿後’), ‘中’ (‘during ...’, ‘within...’, e.g., ‘作業中’), and ‘下’ (‘under... situation’, ‘beneath...’, e.g., ‘狀況下’), at the end of the trigrams, or ‘將’ (‘will’) at the beginning of the trigrams (e.g., ‘將下降’). To resolve such errors, it would be a good idea to include a mechanism for removing n-grams that contains such special characters. The collocational strings also contribute a large portion to the spurious words, which are formed by concatenating single-character words and double-character words that frequently co-occurs due to syntactic collocation (e.g., ‘與台灣’, ‘目前正’).

The spurious quadragrams, as shown in the last row, consist mainly of collocational strings and ordinary n-grams. Both kinds of spurious words share the same feature that they consist mostly of two bigram words. The part of spurious words corresponding to the ‘O’ column contains many ‘noun+noun’ (or ‘adjective+noun’) pairs. Although they take the form of compound words, they may not be considered sufficiently useful for including such quadragrams into the new word dictionary by the lexicographer (e.g., ‘上週大盤’ (‘the closing stock indices of the last week’)), and thus are considered ‘incorrectly identified’ when the system suggests so. Furthermore, some spurious quadragrams are a substring of longer compound words (e.g., ‘國際標準’ is considered as a substring of ‘國際標準組織’ (ISO)) such that they are not regarded as words. The collocational strings have more syntactic variants, assuming the forms of ‘auxiliary verb+verb’, ‘verb+noun’, ‘noun+verb’, and so on. Therefore, it may not be easy to reject such spurious words without using some syntactic information in the model.

To sum up, a numerical string pre-processor before applying the current approach is probably helpful. In the remaining errors, Type II error is the major source of errors for the current approach (which resulted in worse precision performance in comparison to the recall performance). There is still a small fraction of proper names which could not be resolved completely. Furthermore, to improve the precision of the n-grams, it might be useful to detect the existence of some special characters (such as ‘的’, ‘將’) within the n-grams in order to remove such spurious words. It is also probably useful to include additional syntactic processing steps in the filtering stage, so that quadragram spurious words which consist of two double-character words could be removed.

5.7 Summary on Precision-Recall Maximization by Iterative Integration

An unsupervised method is proposed for extracting Chinese new words from a large text corpus in the above chapters. It provides an architecture for us to improve both the precision and recall rates simultaneously, by iteratively integrating the word segmentation module and the likelihood ratio ranking module, so that they can enhance the discrimination power of each other.

The segmentation task is conducted using an *augmented dictionary*, which contains *potential unknown word candidates* in the input corpus, in addition to known words in the system dictionary.

The set of potential unknown words are progressively refined using the joint character association metric provided by the likelihood ratio ranking module, so that the next segmentation iteration is conducted based on a better set of potential unknown words. The parameters of the ranking module could also be improved by using the progressively refined output word list of the segmentation module to re-assign progressively more appropriate class labels to the corpus n-grams, and thus provide better estimates of the log-likelihood ratios for truncating inappropriate word candidates.

In such an iterative process, the precision is progressively improved through the filtering operations, and the real new words corresponding to the rejected spurious candidates are recalled through re-segmentation. The results thus show increasingly improved performance both in precision and recall through iterations.

It is observed that the system could acquire the new word entries with sufficiently high precision in comparison to other previous works, and, yet, still retain high recall. Since this approach adopts an unsupervised learning approach to extract new words, substantial amount of human processing works can be saved, and no pre-segmented training corpus is required before compiling the new word list, the cost for the proposed approach is thus very low. Based on the observation of this study, it is expected that the proposed framework could provide a good tool for lexicographers to extract Chinese new words from a large text corpus.

The performance of different strategies are summarized in the following table. It is easily seen that the iterative approach, being integrating the information used in both the word segmentation module and the likelihood ranking module, outperforms the non-iterative scheme and the other two models which uses only contextual information or only association features.

n-gram	Models	p (%)	r (%)	WPR	FM
2	WS-only	68.72	78.41	73.57	73.25
	LRRM-only	54.28	90.99	72.63	68.00
	Non-Iterative	73.56	73.90	73.73	73.73
	Iterative	72.39	80.83	76.61	76.38
3	WS-only	29.63	81.36	55.50	43.44
	LRRM-only	33.78	63.01	48.40	43.98
	Non-Iterative	31.90	80.34	56.12	45.66
	Iterative	38.60	87.80	63.20	53.62
4	WS-only	38.96	93.09	66.03	54.93
	LRRM-only	51.17	81.42	66.30	62.84
	Non-Iterative	42.38	93.09	67.74	58.25
	Iterative	56.21	93.82	75.01	70.30

Table 28 Comparison of performance between various models for new word extraction.
(WS-only: Word-segmentation only, LRRM-only: ranking module only,
Non-Iterative: cascading WS and LRRM modules and
truncating the worst 10% words of the segmentation output;
Iterative: Iteratively integrating WS and LRRM modules.)

Chapter 6
Concluding Remarks and Future Research

6.1 Summary and Major Contributions

Traditional English compound word extraction methods rely heavily on simple filtering-and-thresholding methods; in general, a set of features are applied, independently, with a set of thresholds to determine whether a candidate is qualified as a compound word. Since the thresholds are determined without relating to any known performance criterion, such approaches are unable to optimize a pre-specified criterion function of the precision and recall performance.

In Chinese unknown word extraction task, the general non-iterative segmentation-merging-filtering- disambiguation scheme is incapable of recovering the over-merging type of segmentation errors; the merging operation may also introduce many randomly merged segments to the filter; and a separated disambiguation process might be needed for resolving overlapping ambiguities since the potential unknown words do not participate in the word segmentation process. Furthermore, due to the non-iterative nature of this scheme, the segmentation module and the filter module cannot help each other to acquire better performance; most likely, the precision and recall will not be improved simultaneously if the two modules are simply cascaded, since no feedback path is provided to recall the real unknown words corresponding to those rejected candidates.

In this dissertation, two precision-recall maximization techniques are developed for the two major lexicon acquisition tasks for English and Chinese. In the English compound word extraction task, emphases are placed on the design of a classifier to maximize a pre-specified criterion function of precision and recall. A two stage optimization scheme is developed for this purpose; the first optimization stage aims at minimizing the classification error rate, and the second optimization stage focuses on the maximization of user specified criterion functions of precision and recall, starting from the minimum error status. In particular, a Gaussian mixture model is adopted to integrate commonly used word association features as well as parts of speech information jointly, with inter-feature correlation taken into account; furthermore, a non-linear adaptive learning method, which adjusts the statistical parameters of the minimum error classifier, is proposed in this task. The main philosophy in this kind of optimization technique is to achieve the best performance by jointly using the most discriminative features, jointly considering the feature correlation, estimating and learning the classifier parameters to satisfy the criterion function.

In the Chinese unknown word extraction task, where the classifier is not the only module of the system, the focus is placed on the integration of the features among different modules through an iterative process so as to achieve the best performance with all available features and information in all modules; the iterative process also provides us with the capability to recover certain errors which prevent a new word from being selected in earlier iterations. In this task, we not only try to improve the precision rate by improving the classifier parameters (and the parameters for the segmentation module) but also try to recover the correct word if the the segmentation module incorrectly identified a word in a previous iteration; an iterative method is thus developed for these purposes. In this manner, the *precision* is improved by using the classifier (or filter) to filter out inappropriate candidates in the segmentation output, and the *recall* is improved by recovering those real words corresponding to such inappropriate candidates using the global contextual constraints. Both precision and recall can thus be seen to increase monotonically without sacrificing precision for recall or *vice versa*, in contrast to conventional filtering approaches. The main philosophy that deserves appreciation in such an optimization technique is to *recover errors* that were made in previous iterations by using other information, in addition to improving the classifier itself (and other system modules) by adjusting the parameters.

6.2 Future Works

Although I had identified several technical problems in those two extraction tasks, and proposed our methods to attack such problems, there is still other room for further improvement. For instance, it is likely that we could apply the iterative approach, which was used for the Chinese unknown word extraction task, to the English compound word extraction task so that we could recover certain errors by using contextual information, and thus further improve the recall performance.

In the Chinese unknown word extraction task, the contextual constraints and other association metrics are integrated by iteratively applying them for extracting likely words and removing unlikely word candidates. However, it would be more elegant if such information can be closely integrated by combining both the contextual constraints and other global character association measures into the same language model, so that the refined language model for word segmentation directly simulate lexicographers' sense of word formation more closely. We can even include some known lexical and syntactic constraints on Chinese lexicon in a probabilistic framework, such as including 'the probability for a word to contain a 'de' marker' (which should be close to zero) in the model. Such integration will then make the segmentation model even closer to the Chinese word formation rules, and thus will allows us to remove strings such as '的人' in a more elegant way, without resorting to special processing.

Once such integration is made so that the segmentation and the classification tasks are integrated into one single language model, it is likely that we could apply the two-stage optimization scheme to this integrated language model. Such issues will be left as our future research.