AUTOMATIC KEY TERM EXTRACTION FROM SPOKEN COURSE LECTURES USING BRANCHING ENTROPY AND PROSODIC/SEMANTIC FEATURES

Yun-Nung Chen, Yu Huang, Sheng-Yi Kong, and Lin-Shan Lee

Graduate Institute of Computer Science and Information Engineering
National Taiwan University, Taiwan
{swallow0130, mnbv711, anguso}@speech.ee.ntu.edu.tw, lslee@gate.sinica.edu.tw

ABSTRACT

This paper proposes a set of approaches to automatically extract key terms from spoken course lectures including audio signals, ASR transcriptions and slides. We divide the key terms into two types: key phrases and keywords and develop different approaches to extract them in order. We extract key phrases using right/left branching entropy and extract keywords by learning from three sets of features: prosodic features, lexical features and semantic features from Probabilistic Latent Semantic Analysis (PLSA). The learning approaches include an unsupervised method (K-means exemplar) and two supervised ones (AdaBoost and neural network). Very encouraging preliminary results were obtained with a corpus of course lectures, and it is found that all approaches and all sets of features proposed here are useful.

Index Terms— keyword extraction, key phrase extraction, course lectures, PAT tree, entropy, prosody, Probabilistic Latent Semantic Analysis (PLSA), machine learning, K-means

1. INTRODUCTION

With huge quantities of multimedia documents available over the Internet, efficient approaches of indexing, retrieving and browsing these multimedia documents are highly desired. Because all multimedia documents may include audio information that very possibly describes the core concepts of the content, automatic extraction of key terms from such audio information (or spoken documents) will be very useful for the purpose of indexing, retrieving and browsing. This paper proposes a set of approaches for key term extraction from spoken documents, and takes course lectures as the example corpus for experiments. Since life-long learning has became necessary for most people today, and there have been huge quantities of course lectures accessible from the Internet, but it is not easy to efficiently browse across these course lectures and find specific information from them, automatic extraction of key terms from course lectures will be very attractive.

Substantial works have been reported on key term extraction from texts domain [1, 2, 3], but much less works on spoken documents were reported [4, 5], specially using information from audio signals such as prosodic features. Some works have been reported using prosodic and semantic features [6, 7, 8, 9] in summarization of spoken documents, though.

We define our task in this work to be the extraction of key terms from a course lecture corpus including the slides used, the audio signals and their ASR transcriptions. The corpus used actually includes a set of slides completely in English, while the lectures were given in the host language of Mandarin Chinese but with all terminologies produced in the guest language of English. Such a Mandarin/English code-switching style is very common for lectures offered in Taiwan. Therefore, the transcriptions include both English and Mandarin Chinese words, and the key terms can be in English or Mandarin.

In this work, we divide the key terms for course lectures into two types: key phrases (e.g. "hidden Markov model" and "information theory" are key terms but "hidden" or "theory" are not) and keywords (e.g. "perplexity"). We propose to use branching entropy to extract key phrases first, and then use three sets of features and different learning approaches to extract keywords. The former is presented in Section 2 below, while the latter in Section 3 and 4. An earlier version of approaches proposed here has been successfully used in a course lecture system developed at National Taiwan University (NTU), refered to as NTU Virtual Instructor [10], while the approaches presented here are much more delicate and advanced.

2. KEY PHRASE EXTRACTION USING BRANCHING ENTROPY

The purpose here is to extract key phrases such as "hidden Markov model" or "information theory". They are the patterns of two or more words appearing together in the corpus (slides and ASR transcriptions) much more frequently than other segments of words. Here we propose to extract such key phrases using a parameter called branching entropy, and one way to obtain these parameters is to use a data structure for

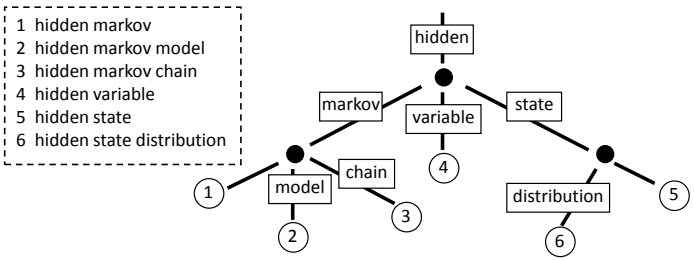

Fig. 1. *A simplified partial list of the word-based PAT tree used to computed the branching entropy*

is hidden Markov model a key term . . . . . .
hidden Markov model a key term . . . . . .
Markov model a key term . . . . . .
model a key term . . . . . .
a key term . . . . . .
key term . . . . . .
term . . . . . .

Fig. 2. *Sistrings of "is hidden Markov model a key term"*

strings of symbols called Patricia tree (PAT tree) [11], with a simplified example is shown in Fig.1. It is a specialized structure based on trie that is used to store strings, and the symbol we used here is the word. We show a simplified partial list of the PAT tree constructed by semi-infinite strings (Sistrings) of sentences in the lecture corpus (slides and transcriptions), where semi-infinite strings are sequences of words starting at any position of the corpus and continuing to the right [12]. Several examples of such semi-infinite strings are listed in Fig.2. We segment each in the corpus (for example, "hidden Markov model...") to its Sistrings ("hidden Markov model...", "Markov model..." and "model...") and use these Sistrings to build the PAT tree. However, some works had been reported on PAT tree-based Chinese key phrase extraction using mutual information [13], but in this work, we proposed another approach, branching entropy, to extract key phrases.

The right branching entropy of a pattern X of two or more words considered is defined as

$$p(x_i) = \frac{f_{x_i}}{f_X}, \tag{1}$$

$$H_r(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i), \tag{2}$$

where X is the pattern of interest (e.g., "hidden Markov"), and x_i is a child pattern of X (e.g., "hidden Markov model", "hidden Markov chain" for "hidden Markov"), and f_X and f_{x_i} are the frequency counts of X and x_i respectively. Thus $p(x_i)$ is the probability of having x_i given X, and $H_r(X)$ is therefore the right branching entropy of X, where n is the number of different child patterns x_i of X.

When a pattern "hidden Markov model" is a key phrase, not only its frequency count is high, but most patterns of "hidden Markov" are all followed by the word "model" (so "hidden Markov" has a low $H_r(X)$), while the patterns of "hidden Markov model" are followed by many different words such as "is", "can", "to", "with"... (so "hidden Markov model" has a high $H_r(X)$). In this way we can use the right branching entropy $H_r(X)$ to identify the right boundary of a key phrase (it is to the right of "model" rather than the right of "Markov" in the above example) by setting thresholds for $H_r(X)$.

Similarly we can construct a "reverse PAT tree" using reverse sequences of words in the sentences of the corpus (e.g. "... model Markov hidden ...") and define a left branching entropy $H_l(\overline{X})$ for each reverse pattern $\overline{X}$. This left branching entropy $H_l(\overline{X})$ can be used in exactly the same way to identify the left boundary of a key phrase (e.g. the left boundary of the phrase "hidden Markov model" is to the left of "hidden" rather than the left of "Markov", because "hidden" is preceded by many different words, while "Markov" is almost always preceded by "hidden".)

The above procedure requires searching through the whole corpus to calculate the left/right branching entropy for every pattern in the corpus. This is quite time consuming if the corpus is large, while PAT tree allows efficient key phrase extraction in this way. In the test, we can compute the average $H_r(X)$ for all X and average $H_l(\overline{X})$ for all $\overline{X}$, and then extract patterns X whose $H_r(X)$ and $H_l(\overline{X})$ are both higher than the average values to be the key phrases.

3. FEATURE SETS FOR KEYWORD EXTRACTION

Three different sets of features are used here: prosodic features, lexical features and semantic features. They are summarized here in this section.

3.1. Prosodic Features

Substantial works demonstrated that prosodic information is useful for summarization of spoken documents, which implied that prosodic features can help extract salient sentences. It was argued that prosodic features in conference lectures are less useful because of the speaker variability [6]. Since the course lectures considered here usually consist of speech produced by a single instructor, the prosodic features may help. It was also claimed that presenters usually use prosodic variation, changes in pitch, intensity and speaking rate for tagging important contents in their speech [7].

In this work, the prosodic features were obtained from the phonetic units segmented by HTK forced alignment [14]. For each term, only the prosodic features for it when it was produced at the first time in the lectures were used. A total of twelve prosodic features was used and presented below.

3.1.1. Duration Related Features

Because different phonetic units have quite different durations, we first compute the average duration of each phonetic unit using the whole course lectures. For each phonetic unit

in a term, we then normalize its duration by its average value. Finally, for each term, we use the maximum, minimum, mean and range of the normalized values for its component units as the four features for the term.

3.1.2. Pitch Related Features

We use ESPS [15] to extract F0 features for frames of each term from the audio data. To avoid discontinuity of pitch contours, we use conventional approaches to smooth them [16]. We then take the maximum, minimum, mean and range from the frames of each term as its four features.

3.1.3. Energy Related Features

For each frame, we take the value of the 0-th cepstral coefficient as the energy. The maximum, minimum, mean and range are then extracted from the frames of each term as its four features.

3.2. Lexical Features

We extract lexical features from both the slides and the transcriptions. These features are presented below.

3.2.1. TF-IDF

TF-IDF usually represents the significance of a term. We segment all sentences in the transcriptions to each slide and take the slide as the document and compute three features, including term frequency (TF), inverse document frequency (IDF), and TF-IDF.

3.2.2. Left Context Variation

We extract two features from the transcriptions based on the left context variation of each term. Our assumption is that the number of different words appearing on the left context of a key term is limited, such as "on", "using", "of" and "is", while this number for a normal term is usually much larger. Therefore we define a left context variation feature lcv_i to be the number of different words appearing to the left of the term t_i in the transcriptions. In addition, this feature lcv_i naturally have to do with the term frequency tf_i of the term t_i, so we further normalize lcv_i by tf_i to obtain a normalized feature $lcvn_i$,

3.2.3. Part of Speech (PoS)

Since verbs, nouns and adjectives are more likely to be key terms, we assign different feature values to each PoS tag [17]. Some terms that can not be tagged are labeled by another value.

3.3. Semantic Features by Probabilistic Latent Semantic Analysis (PLSA)

Probabilistic Latent Semantic Analysis (PLSA) [18] is used to analyze the semantics of documents based on the latent topics. It analyzes a set of documents $\{d_j, j = 1, 2, ..., M\}$ and all terms $\{t_i, i = 1, 2, ..., L\}$ they include by defining a set of latent topic variables, $\{T_k, k = 1, 2, ..., K\}$, to characterize the term-document co-occurrence relationships. The probability of a document d_j generating a term t_i can be parameterized by

$$P(t_i|d_j) = \sum_{k=1}^{K} P(t_i|T_k)P(T_k|d_j). \quad (3)$$

The PLSA model can be optimized with EM algorithm by maximizing a likelihood function [18]. The semantic features obtained from PLSA are presented below.

3.3.1. Latent Topic Probabilities (LTP)

We can evaluate the probabilities for each latent topic T_k given each term t_i, $P(T_k|t_i)$, from the parameters of the PLSA model as

$$P(T_k|t_i) = \frac{P(t_i|T_k)P(T_k)}{P(t_i)}, \quad (4)$$

where $P(t_i)$ can be obtained from a large corpus, and $P(T_k)$ can be estimated based on $P(T_k|d_j)$ in a large corpus. We then compute the mean, variance, standard deviation, variance normalized by mean, and standard deviation normalized by mean for these probabilities $P(T_k|t_i)$ for different latent topics given a term t_i.

3.3.2. Latent Topic Significance (LTS)

Latent Topic Significance (LTS) for a given term t_i with respect to a topic T_k is defined [8].

$$\mathrm{LTS}_{t_i}(T_k) = \frac{\sum_{d_j \in D} n(t_i, d_j)P(T_k|d_j)}{\sum_{d_j \in D} n(t_i, d_j)[1 - P(T_k|d_j)]}, \quad (5)$$

where $n(t_i, d_j)$ is the occurrence count of term t_i in a document d_j. In the numerator of equation (5), the count of the given term t_i in each document d_j, $n(t_i, d_j)$, is weighted by the likelihood that the given topic T_k is addressed by the document d_j, $P(T_k|d_j)$, and then summed over all documents d_j in the training corpus. Therefore the numerator is the total count of the given term t_i used in the given topic T_k over the whole training corpus, as estimated by PLSA model. The denominator is very similar except for latent topics other than T_k, so $P(T_k|d_j)$ is replaced by $[1 - P(T_k|d_j)]$. Same as Latent Topic Probability, we similarly compute five features for each latent topic T_k given a term t_i.

3.3.3. Latent Topic Entropy (LTE)

Latent Topic Entropy (LTE), $\mathrm{LTE}(t_i)$, for a given term t_i can be calculated in equation (6) from the distribution $P(T_k|t_i)$ estimated in equation (4) [9],

$$\mathrm{LTE}(t_i) = -\sum_{k=1}^{K} P(T_k|t_i) \log P(T_k|t_i). \quad (6)$$

Lower $\mathrm{LTE}(t_i)$ indicates that the distribution of $P(T_k|t_i)$ is more focused on fewer latent topics and therefore t_i carries more topical information and is more likely to be a key term.

Table 1. *All features used in this work*

	Feature Name	Feature Description
Prosodic Features	Duration I	maximum of normalized duration
	Duration II	minimum of normalized duration
	Duration III	mean of normalized duration
	Duration IV	range of normalized duration
	Pitch I	F0's maximum value
	Pitch II	F0's minimum value
	Pitch III	F0's mean value
	Pitch IV	F0's range
	Energ I	maximum energy value
	Energy II	mimimum energy value
	Energy III	mean energy value
	Energy IV	range of energy value
Lexical Features	TF	tf_i
	IDF	idf_i
	TF-IDF	$tfidf_i$
	Left Context	lcv_i
	Left Context Nor	$lcvn_i$
	PoS	the PoS tags
Semantic Features	LTP I	variance of LTP
	LTP II	standard deviation of LTP
	LTP III	mean of LTP
	LTP IV	LTP I / LTP III
	LTP V	LTP II / LTP III
	LTS I	variance of LTS
	LTS II	standard deviation of LTS
	LTS III	mean of LTS
	LTS IV	LTS I / LTS III
	LTS V	LTS II / LTS III
	LTE	term entropy for latent topics

4. LEARNING METHODS FOR KEYWORD EXTRACTION

4.1. Unsupervised Learning

We use $\mathrm{LTS}_{t_i}(T_k)$ in equation (7) to construct a feature vector for each term t_i,

$$v_i = (\mathrm{LTS}_{t_i}(T_1), \mathrm{LTS}_{t_i}(T_2), ..., \mathrm{LTS}_{t_i}(T_K)), \quad (7)$$

where K is the number of latent topics. Therefore, v_i represents the position of the term in the latent semantic space.

We use the K-means algorithm to cluster the terms based the above vectors v_i and extract all exemplars of the clusters as the key terms. An exemplar is the vector closest to the median of all vectors in the same cluster. We assume that terms close to a key term in the latent semantic space usually form a cluster, and the key term is close to the median of the cluster [19].

4.2. Supervised Learning

We use all the features in Section 3 and two learning methods to train the classifiers, the AdaBoost and neural network.

4.2.1. AdaBoost

AdaBoost (adaptive boosting) was proposed to obtain a highly accurate classifier by combining many basic classifiers [20]. Given a training set $\{x_n, y_n\}_{n=1}^{N}$, where x_n is the feature vector of a training sample, y_n is the desired label ($+1$ for key term and -1 for non-key term), and N is the total number of training terms. We use decision stump as the basic hypothesis $h_{s,i,\theta}$.

$$h_{s,i,\theta}(x) = \mathrm{sign}\,(s \cdot (x)_i - \theta), \quad (8)$$

where x is the feature vector, $(x)_i$ is the i-th component of x, $i \in \{1, 2, \ldots, d\}$, $s \in \{+1, -1\}$, and $\theta \in \mathbb{R}$ is a threshold.

Initially, we assign the same weights to all decision stumps ($u_n = \frac{1}{N}$ for all n) in equation (9). In each iteration l, we then choose the best hypothesis h_l in equation (9), compute the confidence weight α_l in equation (10) for it, and re-estimate u_n as in equation (11),

$$h_l = \arg\min_{h_{s,i,\theta}} \sum_{n=1}^{N} u_n \cdot [\![y_n \neq h_{s,i,\theta}(x_n)]\!], \quad (9)$$

$$\alpha_l = \frac{1}{2} \ln \frac{1 - \epsilon_l}{\epsilon_l}, \quad (10)$$

$$u'_n = u_n \cdot \exp\left(-\alpha_l y_n h_l(x_n)\right), \quad (11)$$

where ϵ_l is the weighted error for the iteration l. After many iterations, we can get combined hypothesis H, which is parameterized as

$$H(x) = \mathrm{sign}\,(\sum_{l=1}^{T} \alpha_l h_l(x)), \quad (12)$$

where T is the total number of iterations.

4.2.2. Neural Network

We implement the backpropagation algorithm for 3-layered neural network (b-J-1) with tanh-type neurons [21], in which b is dimension of the feature vectors, and J is the number of neurons in the second layer. Stochastic gradient descent is used for many iterations to find the final hypothesis that minimizes the error of the training data.

5. EXPERIMENTS

5.1. Experimental Setup

The corpus used in this research is the lectures for a course on Digital Speech Processing. The slides used are completely in English, while the lectures were given in the host language of Mandarin Chinese but with almost all terminologies produced in the guest language of English. There is a total of 17 chapters with 196 pages of slides, while the lecture is 45.2 hours long. The two acoustic models for Mandarin and English were obtained from the ASTMIC corpus and the Sinica Taiwan English corpus respectively, and were adapted by 25.2 minutes corpus from the target speaker (the course instructor). The language model was trained by two other courses offered by the same instructor and adapted by the course slides. The accuracies for the ASR transcriptions are 78.15% for Mandarin characters, 53.44% for English characters, and 76.26% for overall. The poor English accuracy is apparently due to the small quality of adaptation data but can be compensated by slides information. We set the number of topics for PLSA to be 25, a reasonable number of topics for a course of 17 chapters.

For unsupervised learning by K-means exemplars, we set the number of clusters to be the number of reference key terms discussed below but subtracting the number of key phrase we extracted first. For supervised learning, we use 3-fold cross validation to measure the performance of the classifiers.

5.2. Generating the Reference Key Term List

All people, even if knowing every well the content of the course lectures, may have quite different opinions regarding which word (or phrase) is a key term for the lectures. This is why identifying the reference key terms to be used in this research itself is difficult. For this purpose, we recruited 61 students who had taken the course as subjects to annotate the key terms for the corpus. Since different subjects annotated quite different sets of key terms of different numbers, we assign a score proportional to $\frac{1}{N}$ to a term if it is annotated by a subject who selected a total of N key terms. In this way when a subject annotates less key terms, each of these annotated key terms receives a higher score. We then sort the terms by their total scores assigned by the 61 subjects, and select the top N_0 of them as the reference list, where N_0 is the integer closest to the average of N for all subjects.

A total of 154 key terms (including 59 keyphrase and 95 keywords) were generated as the reference key term list in this way. Some examples of such reference key terms included "language model", "speech recognition", "name entity" (key phrases), "LVCSR", "n-gram" and "entropy" (keywords). Given the reference key term list generated above, annotators achieved an average precision of 66.13%, an average recall of 90.30% and an average F-measure of 76.37%.

5.3. Evaluation Results

We extracted key phrases and keywords separately using the approaches presented above. We used precision, recall and F-measure (equal weight for precision and recall) to evaluate the proposed approaches.

5.3.1. Results of Key Phrase and Keyword Extraction

The results using manual transcriptions and ASR transcriptions are respectively listed in Table 2. We can find that the ASR results are worse than those for manual ones, but they also performed reasonably good. For ASR results, the results for key phrase extraction using branching entropy are reasonable (an F-measure of 68.09%) listed in section 3 of Table 2. The results for keywords are much lower as listed in section 4 of the table. For unsupervised learning, the baseline to be compared is the one using conventional TF-IDF scores after stop word removal and PoS filtering. Here an F-measure of 39.52% was achieved with the proposed K-means exemplar, which is much higher than the baseline though. On the other hand, supervised learning methods offered better results. AdaBoost achieved an F-measure of 49.44%, and neural network (NN) gave 56.55%. Clearly it is much more difficult to identify a single word to be a keyword, and more effort and better approaches are still needed in the future.

Table 2. *Performance of extraction of the two types of key terms: key phrases and keywords using manual transcriptions or ASR transcriptions (%)*

	Type (Num.)	Approach		Precision	Recall	F1
Manual	1. key phrase (59)	Branching Entropy		**59.26**	**81.36**	**68.57**
	2. keyword (95)	U	I	43.84	33.68	38.10
			II	**52.05**	**40.00**	**45.24**
		S	III	54.63	**62.11**	58.13
			IV	**75.68**	58.95	**66.27**
ASR	3. key phrase (59)	Branching Entropy		**58.54**	**81.36**	**68.09**
	4. keyword (95)	U	I	26.39	20.00	22.75
			II	**45.83**	**34.74**	**39.52**
		S	III	44.90	**55.00**	49.44
			IV	**63.08**	51.25	**56.55**

keyword I: baseline TF-IDF, unsupervised (U)
keyword II: K-means exemplar, unsupervised (U)
keyword III: AdaBoost, supervised (S)
keyword IV: neural network, supervised (S)

5.3.2. Feature Effectiveness

We then take neural network to extract keywords from ASR transcriptions as an example since it performed the best in Table 2 to show that all features we extracted here are useful, with results listed in Table 3. Rows (a)(b)(c) show each set of prosodic (Pr), lexical (Lx) or semantic (Sm) features alone

is useful, giving an F-measure ranging from about 20% to about 42%. Row (d) indicates intergrating prosodic and lexical (Pr+Lx) features is better than using either one alone in rows (a) and (b), so the two sets of features are additive. Row (e) shows using semantic (Sm) features in addition helped significantly. Therefore, all the three sets of features are useful.

Table 3. *Keyword extraction (not including key phrase) using different sets of features (%)*

	Features	Precision	Recall	F-measure
(a)	Pr	21.92	19.75	20.78
(b)	Lx	33.57	**59.26**	42.86
(c)	Sm	37.80	33.70	35.63
(d)	Pr+Lx	48.15	48.15	48.15
(e)	Pr+Lx+Sm	**63.08**	51.25	**56.55**

Pr: Prosodic, Lx: Lexical, Sm: Semantic

5.3.3. Overall Results on Manual and ASR Transcriptions

Finally, the overall results for both key phrases and keywords by combining the results in Table 2 are listed in Table 4. The baseline to be compared is the conventional method, of using TF-IDF without considering the key phrases as a different type of key terms. From this table, we find that using branching entropy to extract key phrases is very useful for both manual and ASR transcriptions. Since it improved the F-measure from 23.38% to 51.95% (manual transcriptions) or 43.51% (ASR transcriptions). In other word, once a pattern (e.g. "information theory") is extracted by the branching entropy, the probability that it is a key term becomes high. On the other hand, the K-means exemplar approach proposed here is also reasonably good, considering the fact that it didn't use annotations from the subjects. The best result is the supervised learning using neural network, which offered an overall F-measure of 67.31% for manual transcriptions and 62.70% for ASR transcriptions after combining with the key phrases from branching entropy.

Table 4. *Overall performance (for key phrase from branching entropy plus keywords) using different approaches from manual or ASR transcriptions (%)*

		Approach	Precision	Recall	F-measure
Manual		Baseline	23.38	23.38	23.38
	U	TF-IDF	51.95	51.95	51.95
		Exemplar	**55.84**	**55.84**	**55.84**
	S	AdaBoost	56.61	**69.48**	62.39
		NN	**67.10**	67.53	**67.31**
ASR		Baseline	20.78	20.78	20.78
	U	TF-IDF	43.51	43.51	43.51
		Exemplar	**52.60**	**52.60**	**52.60**
	S	AdaBoost	51.11	**66.19**	57.68
		NN	**60.61**	64.94	**62.70**

U: unsupervised, S: supervised

6. CONCLUSIONS

In this paper we propose a set of unsupervised and supervised approaches for key term extraction from spoken documents, and use a corpus of course lectures for experiments. We divide the key terms into two types: key phrases and keywords, and develop different approaches to extract them in order. Very encouraging results were obtained in the experiments.

7. REFERENCES

[1] A. Hulth and et al, "Automatic keyword extraction using domain knowledge," in *Computational Linguistics and Intelligent Text Processing*, 2004.

[2] Y.H. Kerner, Z. Gross, A. Masa, "Automatic extraction and learning of keyphrases from scientific articles," in *Computational Linguistics and Intelligent Text Processing*, 2005.

[3] P. Turney, "Learning algorithms for keyphrase extraction," in *Information Retrival*, 1999.

[4] F. Liu, F. Liu, Y. Liu, "Automatic keyword extraction for the meeting corpus using supervised approach and bigram expansion," in *SLT*, 2008.

[5] F. Liu and et al, "Unsupervised approaches for automatic keyword extraction using meeting transcripts," in *NAACL*, 2009.

[6] J.J. Zhang, H.Y. Chan, P. Fung, "Improving lecture speech summarization using Rhetorical Information," in *ASRU*, 2007.

[7] J. Hirschberg, "Communication and prosody: functional aspects of prosody," in *Speech Communication*, 2002.

[8] S. Knog, L. Lee, "Improved summarization of chinese spoken documents by Probabilistic latent semantic analysis (PLSA) with further analysis and integrated scoring," in *SLT*, 2006.

[9] S. Knog, L. Lee, "Improved spoken document summarization using probabilistic latent semantic analysis (PLSA)," in *ICASSP*, 2006.

[10] S. Kong, M. Wu, C. Lin, Y. Fu, L. Lee, "Learning on demand - course lecture distillation by information extraction and semantic structuring for spoken documents," in *ICASSP*, 2009. *System available at http://speech.ee.ntu.edu.tw/˜RA/lecture/.*

[11] D.R. Morrison, "PATRICIA-practical algorithm to retrieve information coded in alphanumeric," in *Journal of ACM*, 1968.

[12] D.E. Knuth, "The art of computer programming: sorting and searching," Vol. 3. Addison-Wesley, Mass, 1973.

[13] T. Ong, "Updateable PAT-tree approach to Chinese key phrase extraction using mutual information: a linguistic foundation for knowledge management," in *Second Asian Digital Library Conference*, 1999.

[14] S. Young and et al, "The HTK Book (for HTK Version 3.0)," Cambridge University, 2000.

[15] Entropic, Inc., "ESPS/waves+ with EnSig 5.3 Release Notes."

[16] W. Lin, "Tone recognition for fluent mandarin speech and its application on large vocabulary recognition," in *Master's thesis of NTU*, 2004.

[17] H. Schmid, "Probabilistic part-of-speech tagging using decision trees," in *New Methods in Language Processing,* 1994.

[18] T. Hofmann, "Probabilistic latent semantic analysis," in *University in AI*, 1999.

[19] Z. Liu, P. Li, Y. Zheng, M. Sun, "Clustering to find exemplar terms for keyphrase extraction," in *ACL and AFNLP*, 2009.

[20] Y. Freund, R.E. Schapire, "Experiments with a new boosting algorithm," in *ICML*, 1996.

[21] S. Haykin, "Neural networks: a comprehensive foundation," 3rd edition, 2008.