Automatic Closed Caption Alignment

Based on Speech Recognition Transcripts

By Chih-wei Huang
cwh@ee.columbia.edu

Abstract

Time lagging problem is needed to be solved when using closed caption as a source of features in video indexing. One of solution is aligning closed caption to transcripts then take the time code from transcripts as new references for closed caption. This paper describes a work using limited resources to generate decent alignment results.

1. Introduction

In video processing, both video and audio dimensions of content are intuitively helpful. Recently, lots of video features are applied for better performance, but research about valuable audio features, also believed containing significant information, is relatively much less then video ones. We believe that the speech transcript is a feature carrying most of semantic information among lots of features produced from audio streams. Instead of taking advantage of transcripts from machine, transcripts are also available from human especially on TV programs with more reliability in many dimensions, such as names, new terms, and accuracy. Moreover, special markers and punctuation could be valuable references also. These close captions, available in almost all programs, however, usually suffer lag problems constraining its application in video indexing especially when the program is broadcasted lively. In order to provide a good feature for video processing, the project solves the lag problem in closed caption of live news programs – by doing alignment.

In this project, two kinds of word streams, automatic speech recognition (ASR) output and closed caption (CC), from the same spoken document are put to alignment. After that, CC words are affiliated with new time codes from corresponding ASR words. There are two main contributions in this paper: 1) Proving the possibility using the speech recognizer embedded in Windows XP system to do alignment even though it is capable of mediocre performance only, and the free SDK on the official site [7] provides enough power to handle what the work needs. This is valuable for other groups or individuals to replicate the alignment work. 2) Fusing temporal and structural attributes in alignment process based on classic dynamic programming algorithm since using simply the word identity comparison method cannot handle the entire document especially on commercial and fast spoken parts. The improved scoring system indeed leads to better results. At the end, a modified version of the alignment system is applied to TREC 2003 corpus and shows it is a non-trivial work. With the cue word extracting approach near story

boundaries, the cue word sets extracted from closed caption of TRECVID 2003 are helpful according to some ongoing tests [8].

A related work is presented in [1] by CMU. They use aligned closed caption as the text source for story segmentation. To do alignment, they use dynamic programming based on the degree of initial sub-string match, and mention that even for very low recognition accuracy; the alignment provides sufficiently accurate timing information, but no further details are addressed. However, some issues such as if commercials are included in alignment, if any data other than words are used as support, and etc. are found important according to experiments in this paper.

In [3], on the contrary to our work, AT&T Lab uses alignment to provide time codes of CC to nearly perfect manual transcripts. They utilize transcripts posted on the websites of some program providers, usually without time information, as the final reference for indexing. Since their alignment work is based on two streams with very high word accuracy, it is easier to have good performance than our work using erroneous ASR output to cover more general cases without manual transcripts.

2. Automatic Closed Caption Alignment System

2.1 System Overview

The automatic closed caption alignment system contains two main parts, speech recognizer and alignment engine. The recognizer takes speech audio of the target video as input and generates transcription with time code for each word as output. After recognition, alignment engine uses original closed caption and outputs of recognizer to perform alignment algorithm, which reassign time code for each word in closed caption to proper time spot that the word was actually read in video streams.

2.2 Speech Recognizer

The ideal recognizer is with very high accuracy then we do not have to use CC. Presently, however, the recognizers with enough accuracy to beat closed captions are available in very few labs only or cost a lot of money. Even with sufficient accuracy, the power of special markers like “>>>” and “>>” in CC cannot be replaced. Therefore, using not-perfect recognizer combining CC alignment is one of the best ways to take advantage of CC to compensate the lack of inaccurate ASR. After series of survey, Microsoft speech recognizer with SDK is selected. It is the most convenient one to get under windows platform because it is already in it. After the experiments below, its accuracy is proved adequate for alignment. The SDK package can be freely downloaded from the official website, and it is capable of outputting word level transcription and time codes by some additional programming works. Moreover, if the alignment work could be done on this recognizer, it could be done on almost any other one.

The performance of recognizer should be an important factor influencing alignment performance; it must be evaluated as the basic information of this system. The word error rate (WER) is defined as combination of insertion, deletion, and substitution rates. For example, if an audio stream “It is mostly sunny in Central Park” is recognized as “It is almost leading sun near Central Park”. There are three substitutions “almost”, “sun”, and “near”, and one insertion “leading”, so the WER is 4 out of 7 words of original stream, i.e. 4/7 ~57%.

In order to evaluate the recognizer performance, the anchor part of a half-hour-long CNN news video clip, totally 3742 words, was manually transcribed. To do the transcription efficiently, a freeware – Transcriber from LDC (Linguistic Data Consortium) [5] is used.

Table 1 shows the WER of different cases. The WER of MS Speech SDK is 60.44% for 30 min CNN news without commercial. Comparing to CMU Sphinx-III system [6] has ~35% WER for 30 min CNN news in 30 times real time.

Type of Speech Data	Recognition System	Word Error Rate
News Text Spoken in Lab	Sphinx-II	~10% - ~17%
Evening News (30 min)	Sphinx-II	~50%
	Sphinx-III (30xRT)	~35%
	Sphinx-III (300xRT)	~24%
Commercials	Sphinx-III	~85%
CNN news (30 min)	MS Speech SDK	~60%

Table 1: Word error rates in different cases.

In alignment, a word in closed caption might be correctly, or closely, assigned a time code even its corresponding word in transcription is erroneously recognized. Obviously, higher recognition accuracy results in higher alignment accuracy, because two identical words in closed caption and speech have strong probability to occur at the same time spot. If erroneously recognized words can be located and matched to closed captions appropriately along time axis, word error rate could be tolerated to some extent.

Some advanced issues are also taken into considerations. The first one is training capability. In speech recognition, training is the most efficient way to boost up the accuracy. If acoustic and linguistic models are more specific for the corpus type used, better transcription results could be generated. The default setting for the recognizer embedded in Windows XP is for dictation with carefully speaking, not for spontaneous speech inputs like anchors do in news programs. The second, a one-hour-long spoken document is going to be treated as input, so the recognizer should be able to handle this input size and finish the recognition in reasonable time length. Therefore, a testing program was implemented and the outcome was quite satisfying. The recognizer may self-training by iteratively recognizing the same corpus, and the improvement is very obvious between the first and the second time applying to a new sort of audio. The time consuming is also acceptable in this recognizer, it needs only even a little bit less real-time on machine with two Intel Xeon 1.8 GHz CPU and 1GB of RAM.

2.3 Alignment Algorithm

2.3.1 Dynamic Programming

There are two word streams, ASR and CC, in the matching work. The kernel of alignment program is based on dynamic programming [2] technique. This best path finding procedure iterates word-by-word along both streams and ends when both streams reach the last words. It is very similar to determining the minimum number of word edit required to convert one string into another [3][4]. The differences are using words as basic elements rather than characters and using distance, carefully measured floating numbers, as score measurement rather than integers, the count of word edit. During the alignment, the score of a word depends only on possible paths to it. The paths could be a match, substitution, insertion, or deletion from previous words, and scores are the sum of previous scores and transition scores to current ones.

If we use distance instead of score for alignment, and the distance matrix is *D*, *i* is the index of ASR words, and *j* is the index of CC words. Therefore, *D(i, j)* is the minimum distance accumulated from the first ASR and CC words to the *i*th ASR word and the *j*th CC word. Because both word streams can only iterate forward word-by-word and the nature of dynamic programming that the minimum distance of one state (word) can be evaluated from possible states right before it only, *D(i, j)* can be represented as below:

$$D(i,\, j) = \min\begin{cases} D(i-1,\, j) + ins(i,\, j) \\ D(i,\, j-1) + del(i,\, j) \\ D(i-1,\, j-1) + ms(i,\, j) \end{cases} \tag{1}$$

where *ins(i,j)*, *del(i,j)*, and *ms(i,j)* are one-step transition distances of three possible cases, insertion, deletion, and match or substitution respectively. When the insertion case is chosen as minimum distance path, a word need to be inserted into CC stream to match ASR stream, but we keep CC unchanged and just insert a blank in that time spot. In the same way, when deletion is the case, a word in CC stream should be deleted; we leave that word without a matching ASR word. Note that insertion cases are much more possible than deletion ones because there are always misses in CC. When the remaining one is the case, a match or a substitute of words should be the best path here.

2.3.2 Scoring

Several scoring sets are tested in this work. The first kind of set tested is the simplest and straight forward one with binary scores only for each word pair. When perfect match occurs, *ms*(*i, j*) is set to 0. Otherwise, transition distances are set to 1. In my program, the final alignment will be the case with most matched words and unmatched words will follow matched words then insertions as shown in Table 2.

Obviously, the main drawback of this simple scoring set is taking no control of substituted and inserted words. When erroneous sections, all sections other than matched

ones, are long or overlapped with any long missing section in CC, e.g. commercials, the performance will decay seriously. For example, if there is no, or few, CC during commercials, which is often the practical case, and beginning words of the next story section are incorrectly recognized, CC words corresponding to this section might be aligned to the previous story section leading to large alignment errors in a range of several minutes. Consequently, a new scoring set which can maintain the CC structure and it's relation to ASR should be applied to well distribute unmatched words. This is the most important problem I dealt with and the detail will be addressed below.

ASR	CC	Aligned Type
the	THE	Match
fires	FIRE	Match
are	WAS	*Sub*
nearby	STILL	*Sub*
woods	BURNING	*Sub*
is	----------	*Ins*
done	----------	*Ins*
during	----------	*Ins*
an	----------	*Ins*
hours	HOURS	Match
after	AFTER	Match
it	IT	Match
started	STARTED	Match

Table 2: Alignment results using binary scoring only.

2.3.3 Prefix Processing

Except for distributing issue, the determination of match is needed to be revised, too. Since ASR output contains lots of error words with close pronunciation to correct ones, if the scoring set can determine this situation as partial match and assign lesser distance to closely pronounced words between ASR and CC, the result might be improved. Some approaches are tested. The first one is using word edit distance to give two words a match score from 0 to one. Though it is not judging by pronunciation, similarly spelled words are very likely having similar pronunciations. Its computing cost, however, is extremely large when combining with alignment algorithm, because it is a dynamic programming process, too. The data of one-hour-long video stream needs one day to align on a machine with two Intel Xeon 1.8 GHz CPU and 1GB of RAM in Matlab code.

The final approach I used is comparing only prefix of words, and assign a constant score for these partial matched words if their prefixes are identical. Though this approach is simple, it dose matches the nature of language that words start in identical characters might have close pronunciations in the beginning. Therefore, to simplify, the prefix here is the first constant number of characters of words and the match/substitution distance,

ms(*i*, *j*) in Eq.(1), will get a constant value between 0 and 1 when *i*th ASR word and the *j*th CC word are not identical but the first 2 or 3 characters of them are.

Of course, another possible approach is compare ASR and CC directly in phoneme domain, but it is more complicated and needs additional information for word-phoneme transform. This approach was not implemented.

2.3.4 Exponential Decaying Function

In order to maintain the structure of CC, an exponential decaying function depending on time code in CC is included in distance measure during alignment. The word "structure" here means the distribution of CC words along the time axis. For example, words should be very close when the anchor speaks fast, and be very sparse when it is in commercial for its usually being ignored by CC typer. In ASR, however, word gaps are quite uniform because the recognizer generates transcript words continuously no matter what the content of spoken document is. Since the testing CC data also provide time codes for each line of CC, the time code of each word could be estimated by simply using interpolation referencing to lengths of words. After getting the time code for each word, the time gaps between two successive words can be evaluated and used as a parameter in decaying function.

Because of the relative behavior between insertion distance and word time gaps in CC, the exponential function is applied to model it. When the time gap is very small, there is lesser possibility to have an insertion, shown as a blank in results, in CC stream, and the distance should be large. When the time gap is very large, probably occur in commercial parts, many insertions should be allowed between CC words because there are still lots of words in ASR corresponding to this time slot. The new insertion distance with decaying function is in the form of:

$$ins(i, j) = a + \frac{1 - a}{e^{b \times (t_{j+1} - t_j)}} \tag{2}$$

where $0 \le a \le 1$, b is a constant parameter, t_{j+1} and t_j are time codes of j+1th and jth CC words in seconds. When the time gap, $t_{j+1} - t_j$,is small, the exponential term will be close to 1 and *ins*(*i*, *j*) will be close to 1 also, which means here should not allow easy insertions. When the time gap is large, the exponential term will getting small and *ins*(*i*, *j*) will be close to a, which allows insertion more easily.

2.3.5 Closed Caption Distribution Modeling

Another important properties in given data is the relation between two, ASR and CC, streams. Evidently, original times of CC words should have lags in a reasonable range, couple of seconds in general, comparing to ASR words with which CC words should be matched. If we can obtain the lagging likelihood distribution along time axis in advance,

some impossible paths in dynamic programming could be avoided or paths with different possibility could be given proper distance values.

To implement this idea, two experiments are performed. The first one is just constraining the original CC timing in reasonable range, 0 to 10 seconds for example, and then CC words will have very little probabilities to be aligned to ASR words outside of the range.

The advanced one is to actually apply probability distribution function of CC. To get CC distribution, histogram of CC lag times is counted first; then applies a moving average filter to do smoothing. At the end, an additional distance value coming from the pdf will be added to all three possible paths, insertion, deletion, and match of substitution, to the word which is being aligned to express proper distance increases due to less probability to have the CC lag value. The new distance, $D'(i, j)$, of the ith ASR word and the jth CC word is:

$$D'(i, j) = D(i, j) + c \times \left|\log(d \times p(dCS))\right| \tag{3}$$

where $D(i, j)$ is whatever distance value without distribution modeling, which is usually the value after applying binary scoring, prefix processing, and decaying function. c and d are parameters to do adjustment. $p(dCS)$ is the probability of time gap dCS between the ith ASR word and the jth CC word.

2.4 Visualization of Alignment Results

Except for alignment algorithms, a convenient way to demonstrate the aligned and unaligned CC is also put in use. After transform words and its corresponding time into .srt formate, a format used by a freeware, Vobsub, which can attach real-time subtitles to video when it is played on windows media player, we can easily judge if the CC is aligned well. This demonstration is very effective and impressive.

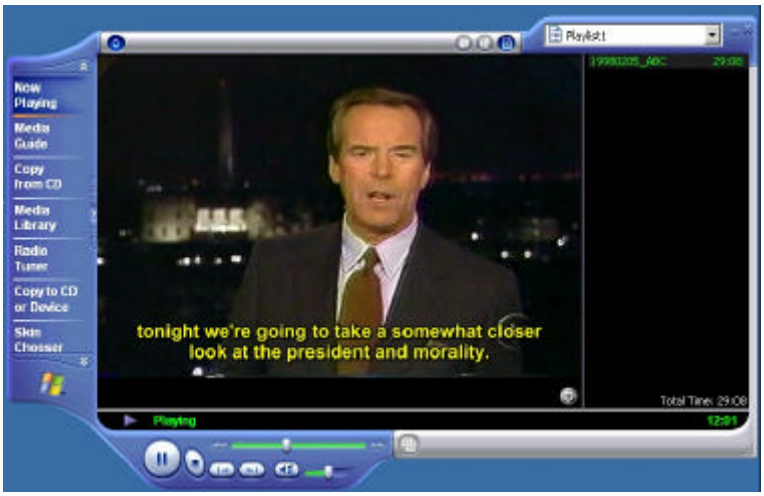

Figure 1: Snapshot of alignment visualization

3. Experimental Results

3.1 Experiment Overview

Labeled testing database is one-hour-long CNN news with CC. After machine recognition, complete word and time information with sentences determined by pauses can be obtained. In CC data, words are listed in lines same as they showed on television and time codes are attached for each line. Interpolation is performed to generate time code for every word.

All alignment results are represented by distributions of time difference between aligned time and actual time of selected words. For video indexing based on news stories might be the future use of this project, and sentence boundaries are also story boundaries for most cases, sentence beginning words are chosen for evaluation. In this project, 376 sentence beginning words in one-hour-long CNN news with time codes are manually marked.

For commercial parts, the most difficult part need to be dealt with, in both ASR and CC data, no additional pre-process applied. It is nearly impossible to kill all commercial automatically and precisely in such noisy ASR transcripts. Though commercial killing is more reliable in this CC corpus, it is still a little bit less than perfect due to lots of different format of commercial boundaries result from different typers. Therefore, the commercial killing algorithm for a series of video clips may not work on another. To generalize my experiment, the original CC content is used no matter how much of commercial is transcribed.

The x-axis in word distribution charts below are time differences in seconds. Positive values are corresponding to cases that words in closed caption are assigned to time spots after they are actually pronounced in video; negative values are on the contrary. Obviously, the more concentration around “zero” of x-axis, the better alignment result it is. If alignment errors are larger than 10 seconds, all of them are represented as +/- 10 second instead. The y-axis is the percentage of selected words in each time slot. Each time slot covers 0.5 second, for example, if there is a bar at 0 in x-axis and 0.53 tall in y-axis, means there is 53% of 376 sentence beginning words are aligned to less then 0.25 seconds before or after the exact time it should be. Besides time difference distribution charts, percentages of aligned words within +/- 0.75, +/- 1.25, +/- 5 seconds, and outside of +/- 10 seconds, mean, and standard deviation are also collected.

3.2 Results

3.2.1 Original Closed Caption

Original CC, just as we see on TV everyday, has several seconds lag from we hear the words. The lags are mainly between 1 to 5 seconds as shown in Figure 2. From Table 3, we can see it has mean delay around 3.7 seconds, small standard deviation, and only 1.6% of words are in the out side of 10 seconds range.

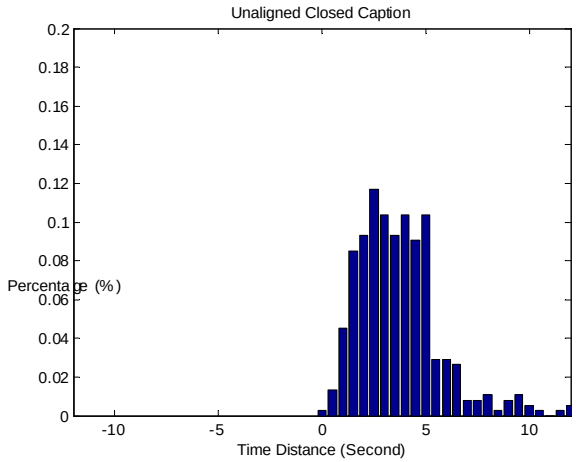

Figure 2: Word distribution chart for original closed caption.

3.2.2 Aligned Closed Caption

The outcome of aligned closed caption is shown in Table 3. Different experiments are separated by double horizontal lines, and the best data in each column are marked as bold and italic type.

The first experiment, treated as the baseline test of all experiments, uses only binary scoring approach. This straight forward approach does generate the result with most matched words but does nothing to unmatched words. Low centralization degree, 50.798% in +/- 1.25 s, and a lot of extreme cases, 22.074%, are noticeable. From Figure 3(a), we can see lots of word aligned to the minus region which illustrates the property shown in Table 2 – substitutions take place before insertions in consecutive unmatched word streams. Thus, unmatched words have a tendency to be aligned earlier, and results in minus values in time distances.

After adding prefix processing to the scoring system, data in the rows of B+P(2) and B+P(3) show better performance than the baseline in all aspects except for the mean values. The number of characters counted as partial match is set as 2 and 3 respectively, and the distance when partial match occur is set as 0.5. According to Figure 3(b), mean values change from -3.4 to higher than 7 because of the reduction of extremely early aligned words while late aligned words remain in almost the same amount.

When apply binary scoring and decaying function to dynamic programming, all statistical data show very significant improvements comparing to the baseline. Parameter *a* in Eq. (2) is set as 0.5 while *b* is set as 0.5, 1, and 2 respectively. ~20% more words have less then +/- 1.25 seconds of alignment errors, and words with extreme errors decrease ~20% of total words. Notice that because of the nature of exponential decaying functions, if parameter *b* in Eq. (2) is getting larger, the decaying behavior increases dramatically, same amount of time distance between two nearby words result in smaller insertion distance, which means easier to insertion, with larger *b*. Thus, with lager b, the closer

words in original will be closer in aligned result; the farer ones will be farer. Some words were pushed too far in these experiments, which can be seen from mean and standard deviation columns from B+DF(0.5)/(1)/(2) cases in Table 3. However, the sensitive functions, i.e. with larger *b*, increase the percentage of words within +/- 1.25 seconds slightly, ~0.7%, because correct aligned words, usually not close to story boundaries, have more probabilities to stay closer. Figure 3(c) is the results of B+DF(0.5) which is consider as the best performed in this part of experiment one for its better mean and standard deviation values.

	Mean (s)	Std (s)	+/- 0.75 s (%)	+/- 1.25 s (%)	+/- 5 s (%)	> +/- 10 s (%)
Original CC	3.712	1.992	1.596	6.117	85.11	1.6
B only (Baseline)	-3.413	59.249	35.904	50.798	73.138	22.074
B+P(2)	7.646	40.82	39.362	56.383	76.33	19.947
B+P(3)	10.336	48.059	38.564	54.521	76.33	19.947
B+DF(0.5)	-0.916	3.8249	50	69.947	92.287	2.66
B+DF(1)	-3.336	24.192	50.266	70.745	92.553	3.457
B+DF(2)	-5.161	42.451	50.532	70.745	92.287	4.255
B+P(2)+DF(0.5)	-0.882	3.769	51.064	71.809	92.287	2.128
B+P(2)+DF(1)	-0.93	4.357	52.128	72.872	93.351	2.926
B+DM	12.06	46.395	37.5	54.255	80.585	15.691
B+P(2)+DM	15.876	55.203	39.894	59.043	79.521	17.553
B+DF(1)+DM	-0.383	2.252	51.064	72.074	95.745	0.532
B+P(2)+DF(0.5)+DM	***-0.338***	2.229	51.33	73.404	95.745	0.532
B+P(2)+DF(1)+DM	-0.366	***2.165***	***51.862***	***73.67***	***96.011***	***0.266***
B+P(2)+DF(1)+DM(1)	-3.358	26.453	20.213	39.362	95.479	1.064

Table 3: Statistical data of experiments. The left most column shows techniques applied for the data on its right side. “B” represents “Binary scoring”, P(*n*) represents prefix processing with *n* characters counted, and DF(*m*) represents decaying function using *m* as it’s *b* parameter in Equation (2). DM without parameter represents distribution modeling with range constraining only. The parameter *l* in DM(*l*) represents parameter *c* in Eq. (3).

Combining binary scoring, prefix processing, and decaying function, results are even better in all aspects, row B+P(2)+DF(1) and B+P(2)+DF(0.5) in Table 3. Remarkably, with the support of prefix processing, mean and standard deviation values in highly sensitive decaying functions catch up with *b*=0.5. Though the difference from Figure 3(c) is not much, it dose show more concentration and less bad alignments in Figure 3(d).

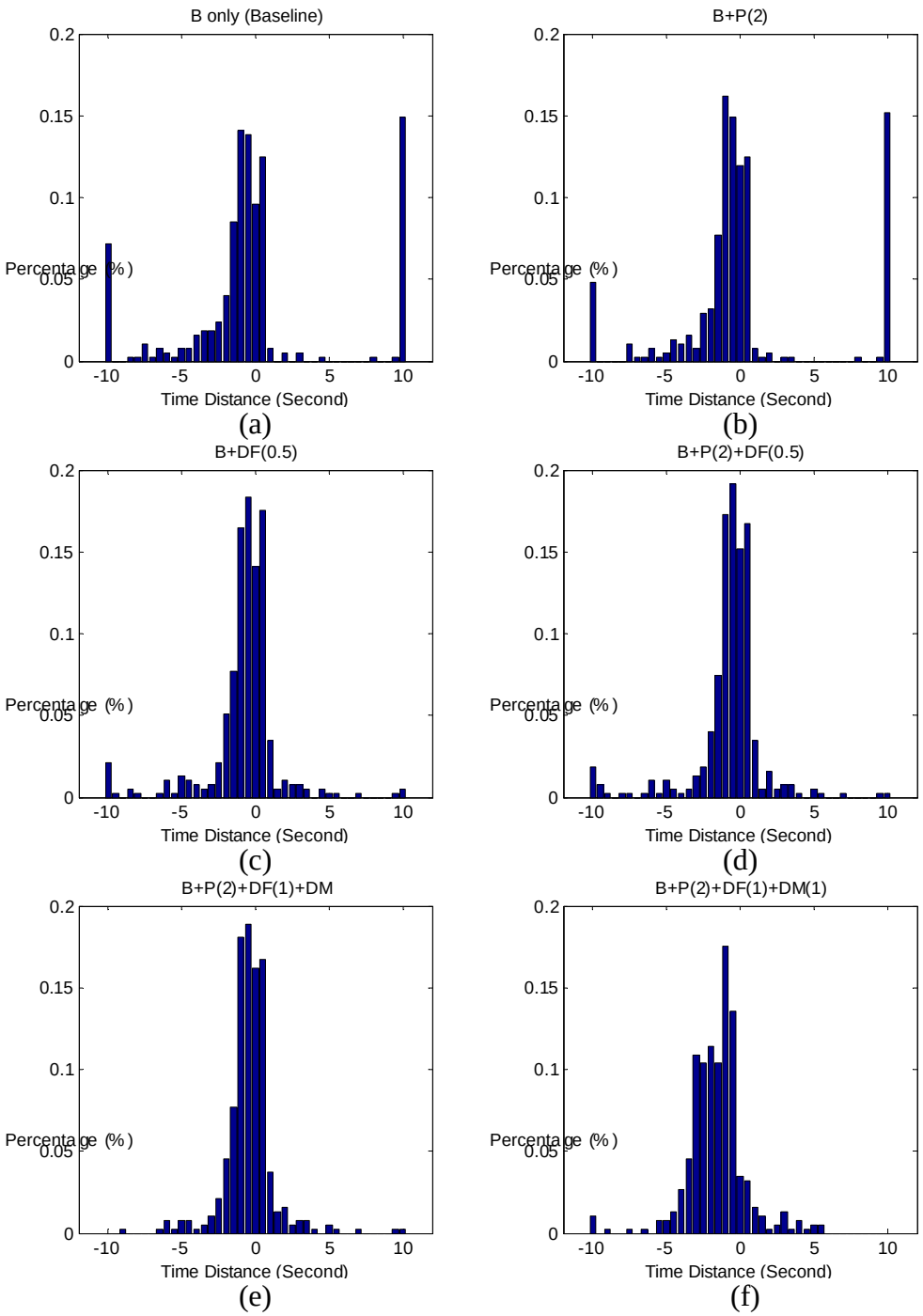

Figure 3: Word distribution charts after aligned in different cases.

When further applying range constraining, 0 to 10 seconds, to limit final alignment path, I find that it only works well with decaying function; otherwise, the improvement is not much. The reason is if the constraining is added to a case with lots of extremely aligned words, like B only or B+P(2), it only has power to force the path to go along the edges of reasonable range which is still probably have around 10 seconds of error. However, adding constraining to the cases with decaying function not only improves all aspect of data but also diminishes the extreme bad cases from 3.457% to 0.532%, comparing row

B+DF(1) to B+DF(1)+DM in Table 3. As I expect, the performance gains more with all scoring techniques as shown in row B+P(2)+DF(1)/DF(0.5)+DM in Table 3 and Figure 3(e). Not like previous cases, the DF(1) case here outperformed DF(0.5) case in all data except ~0.03 second worse on its mean value.

The last experiment is to use Eq. (3) for scoring. Parameter *d* is set to let the pdf value of time slot with maximum pdf to be 1, i.e. result in 0 after log operation to cost the least distance. In my experiment, 2.25 to 2.75 is the most possible time slot of CC lags, and the values of second terms of 7 slots nearing the maximal one in Eq. (3) is under 0.01, which means they would not cost too much distance also. However, from Table 2, the last row is not the best performed one. Figure 3(f) shows the result. The distribution is shifted to the minus part, but not like Figure 3(a), its distribution in minus part is quite flat which should be caused by distribution modeling. The scoring method forces some word to be aligned by distribution probability, not by identity of words especially on the words originally more than 5 seconds away from the actual spoken spot. The logged distance value raises too fast after 5 seconds and make words hard to aligned to where it should have be. Some better functions need to be developed for this problem.

4. Modification for TRECVID 2003 Corpus

The first successful application of this alignment project is for TRECVID 2003 corpus. A modified version of alignment algorithm addressed above is applied on it and results in decent outcome to be provided back to NIST and other groups. Furthermore, combining the cue word extracting algorithm developed on fall 2002, we can even extract cur words near story boundaries from CC to be a feature for indexing. The feature could support the performance to some degree when ASR recognition accuracy is low or the term we want is new and does not exist in dictionary.

The corpus, provided by NIST to TRECVID members for international video indexing evaluation, includes ASR and CC of 250-hour video clips. The ASR transcripts are obviously more accurate than those generated by Microsoft Speech SDK in my project, but CC parts are with word tokens and punctuation marks only, no time information on them. Consequently, the decaying function and distribution modeling, which need time codes in CC, could not be applied here. Besides binary scoring and prefix processing, a new idea to control unmatched words is developed.

The first approach implemented is based on period marks in CC. Because two words belong to the same sentence should not have too large time gap between them, different distance values are assigned to different conditions, in or not in the same sentence, during dynamic programming process. When two words belong to different sentence, the distance of insertion will be smaller to let insertion easier to occur and vice versa. This is the 1.0 version of aligned CC published on the website. Even didn't apply decaying function on it, this version has apparently better accuracy then MS SDK one for it's higher ASR accuracy.

Besides period marks, pause marks and time codes in ASR could be useful in alignment, too. For pause marks, we might model it's relation to the sentence ends or beginnings. For ASR time codes, we might revise distances give to each word to maintain CC structure better especially near commercial boundaries. These two idea are not yet implemented and going to be tested in the future.

5. Conclusion and Future Works

In addition to the words itself, pronunciations, relative time gaps, and original distribution in CC are all have its power in ASR-CC alignment. This paper indeed proves the performance gain of introducing these elements is significant. Because fine tuning and improve the accuracy of ASR system is a really tough work and highly depends on the background of speech, letting aligned CC to compensate recognition errors is efficient and could be done as experiments shown above.

In the future, more corpus should be tested and evaluate their performances to make experiments more general. The TREC 2003 corpus, at this moment, might be the best database to do further researches of this topic. Therefore, a scoring system for dynamic programming which take fully advantage of information provided by TREC should be developed and systematically evaluated.

6. Reference

[1] Alexander G. Hauptmann, Michael J. Witbrock, **Story Segmentation and Detection of Commercials in Broadcast News Video**, Advances in Digital Libraries, 1998

[2] Nye, H., **The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition**, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1984

[3] David C. Gibbon, **Generating Hypermedia Documents from Transcriptions of Television Programs Using Parallel Text Alignment**, Continuous-Media Databases and Applications. Eighth International Workshop on, 23-24 Feb. 1998

[4] R.A. Wagner and M.J. Fischer, **The String-to-string Correction Problem**, *Journal of the ACM*, 21(1):168-173, January 1974

[5] http://www.ldc.upenn.edu/mirror/Transcriber/

[6] http://www.informedia.cs.cmu.edu/dli2/talks/Oct30_99/sld026.htm

[7] http://www.microsoft.com/speech/download/sdk51/

[8] Winston Hsu, Shih-Fu Chang, Chih-Wei Huang, Lyndon Kennedy, Ching-Yung Lin, and Giridharan Iyengar, "Discovery and Fusion of Salient Multi-modal Features towards News Story Segmentation," to appear in SPIE/Electronic Imaging, Jan. 18-22, 2004, San Jose, CA.