Daniel Chai
CS 224N
Final Project

A Trainable Web Page Document Summarizer

(This project was extremely frustrating. I chose a problem domain that interested me (the web), and I wanted to see whether existing techniques would work in this domain. The frustrating thing is, I wrote far more code and spent far more time on this project than for the previous assignments, and but had marginal success. In particular, none of the adjustments I made proved to be fruitful. There were some key lessons learned, but that does not make the frustration any less acute, as I suppose always happens when the performance of a system does not reflect the effort expended. But that's more than enough venting.)

This project involves the summarization of web page documents by extraction of key sentences in the document.

An increasingly interesting problem of NLP is that of document summarization. Document summaries are abstract-like synopses of full text documents. Summaries are extremely useful in allowing users to quickly determine the usefulness of a document. In most domains, such as abstracts for scientific journal articles, the summarization is done by humans; usually the author. Unfortunately, in many domains in which it would be useful, human summarization is not possible or feasible. Recently, however, with the explosion of the web and search engines, automatic text summarization has grown in popularity. A common use of this is in presenting results in web searches. Unfortunately, such summaries tend to be extremely short, often too short to determine if the document is really relevant, and extremely dumb. Most search engines simply present the first few lines of the document. Others show the parts of the document that match the query. In any case, these summaries are usually not too helpful in determining the usefulness of a document on their own. The development of a good automatic document summarizer would thus be very valuable.

This project focused on the summarization of web page documents - web pages that are mostly text. The goal was to see whether existing document summarization techniques could be applied to web page documents. There are many possible applications for this; for example, the presentation of search queries could present automatically generated summaries of documents, which would provide more useful information to the user.

The system was based on the document summarizer developed by (Kupiec et al. 1995). A discussion of their system and how it was adapted for this problem follows.

Kupiec et al Document Summarizer

Sentence matching
The Kupiec system is essentially a modified Naive Bayes classifier. Their training corpus consists of document-summary pairs from a number of scientific journals. Before training, they run a sentence-matching algorithm to determine which sentences in the document correspond to sentences in the summary. Instead of doing sub-sentence information extraction, the system concentrates on sentence selection, so sentences in the training data are marked as being part of the summary or not.

Feature set
The system examines a set of features for each sentence, all of which are discrete:

- **Sentence Length Cut-off Feature:** Short sentences tend to not be included in summaries. Sentences longer than a certain threshold have a value of true, and the others are false.

- **Fixed-Phrase Feature:** Sentences containing or following a number of key phrases tend to be included in summaries. Kupiec et al. use 26 indicator phrases. Sentences containing those phrases, or ones following section heads with key words, have a value of true. All other sentences have a value of false.
- **Paragraph Feature:** Sentences are categorized as being the initial sentence in a paragraph, a medial sentence, or a final sentence.
- **Thematic Word Feature:** The most frequent content words are designated theme words. Sentences are ranked based on the frequency of theme words. This feature is binary, depending on whether a sentence is among the highest ranked sentences.
- **Uppercase Word Feature:** Proper names and acronyms tend to be important. This feature is similar to the previous one, with sentences ranked according to the frequency of uppercase words that appear more than once in the document.

Classifier
For each sentence s, the system determines the probability that it will be included in a summary S, given features F_j, j = 1...k using a Bayes decision rule:

$$P(s \in S \mid F_1, F_2, \ldots F_k) = P(F_1, F_2, \ldots F_k \mid s \in S) \times P(s \in S) \; / \; P(F_1, F_2, \ldots F_k)$$

Assuming that the features are statistically independent, the above becomes:

$$\Pi \, P(F_j \mid s \in S) \, P(s \in S) \; / \; \Pi \, P(F_j).$$

Thus, the system is essentially a naive Bayes classifier. $P(s \in S)$ is constant, and $P(F_j \mid s \in S)$ and $P(F_j)$ are estimated from the training set by counting occurrences.

I was actually surprised at the lack of detail in the cited paper, and the evaluation measures they used. Many of the details of system implemention is not discussed, and it is unclear how exactly their system worked. So many issues had to be worked out, and this is discussed in the implementation notes below. These notes discuss the implementation details of my system, different approaches I attempted, and how it differs from the Kupiec system.

Implementation Notes

Corpus
The first problem was assembling a suitable corpus. I decided ahead of time that I wanted to work with web page documents, so I needed a way of assembling a corpus.

This turned out to be much harder and more time consuming than I anticipated. A big problem was finding a good source for web documents. At first, I examined web sites such as cnn.com, espn.com, mrshowbiz.com and other popular informational sites for web documents. However, when doing manual selection of summary sentences and running the program on these documents, I found that they are typically best summarized by the first few sentences in the document. Thus, the document summarization system could do extremely well by following a trivial selection algorithm – simply selecting the first sentences of the document. This is obviously not an interesting problem, but it turned out that most of the documents on popular web sites were like this – the first few sentences in the document summarized the document. This was one of the lessons I learned, and while it was something that makes sense, it is not something I fully realized until trying to summarize web pages.

To keep the problem non-trivial, I had to look at other sources for web documents. I decided on two domains. One was salon.com, an online magazine that contains longer articles and stories than those found on typical news sites such as the ones mentioned previously. The other was the online journal of a

friend, "Dave." I wanted to see how applicable the algorithms used by Kupiec et al. were on different domains, and I found these to be sufficiently varied. salon.com is typical of online magazines, and the dave data is typical of personal web journals, which are increasing in popularity.

Once I settled on a set of web pages, I developed a program written in perl to strip the web pages of tags and leave the text, one sentence per line, separated by paragraphs.[1] There was a concern as to whether doing so removed any potentially useful information, but it was found that almost the only tag found within the body of the text was <p> paragraph breaks, and this paragraph information remained in the data file.

The next problem was tagging the training data, and marking which sentences belonged in the summary. There were no abstracts for the documents so I could not take Kupiec et al's approach. Instead, I manually went through each document, marking which sentences I judged as belonging in a summary of the document. This was also much more time intensive than anticipated, but there was no other alternative.

Because of the time intensiveness of the process, I was forced to have a limited corpus – six documents each in the salon.com training data and the dave training data. This is considerably less than the corpus used by Kupiec et al., who had 188 documents. Their corpus involved 498 summary sentences. The training sets I assembled had 84 summary sentences (salon data) and 97 summary sentences (dave data). The amount of data is sparse, but there are still a good number of sentences to do training with. The limited amount of training data is unfortunate, but again, given time constraints, it was unavoidable.

For training, the system used the same cross-validation approach as the Kupiec system: documents were selected for testing one at a time, and all other documents were used for training. Results were summed. As they note, this cross-validation technique is the best way to work with a limited data set such as the ones I examined.

From the beginning, it was recognized that this corpus represented a much more difficult problem than the problem domain for the Kupiec system. Their corpus consisted of scientific journal articles, all of which followed certain formatting and content standards. These standards reflected certain common features, which were exploited by the system. It was not expected for these features to be present in the web document data I used. This was an intentional choice - the purpose of the project was to see whether Kupiec et al's methods were widely applicable, or limited to their problem domain.

These training sets are included in the electronic submission, and more information about them can be found the in the submitted README.

Features

My system began by using the same feature set as the Kupiec system, but certain modifications and decisions were made:

- **Sentence Length Cut-off Feature:** Kupiec et al. do not indicate the threshold value used for this feature, although they list 5 words as a possible value. This is the value I chose, so "long" sentences were ones more than 5 words in length. Empirically, this value seemed to work best in eliminating short non-summary sentences.

- **Fixed-Phrase Feature:** While this feature may have been relevant for scientific journal articles, I found that there were no such indicator phrases present in my training data. In both training sets, there were no section headings, and no keywords (such as "in conclusion", or "summary") could be identified. Thus, this feature was not used in this system. It is a feature relevant to scientific journal articles, but extremely difficult to find in web documents in general.

[1] perl expertise and implementation was provided by fellow CS student Eric Mao, not enrolled in CS 224N

- **Paragraph Feature:** In addition to the paragraph position categories Kupiec et al. used, I included a "paragraph alone" category, when a sentence was the only one in the paragraph (delimited by <p> tokens).

- **Thematic Word Feature:** Theme words are defined as being "content words." Kupiec et al. do not indicate how they determined whether a word was a content word or not. For example, words such as "I", "and," "of" and such are clearly not content words. They may have used a list of common non-content words, or they may have manually determined which words were or were not content words.

  In light of the goal of the project, automatic summarization, I did not want to take the second approach, which involves manual analysis. However, there was no list of non-content words available. My solution was to only look at words greater than 3 letters in length. This eliminated most of the common non-content words in the document, and I found experimentally that theme words were never less than 4 letters long. While this did not exclude all extraneous words (such as "than"), it was nevertheless useful. Consequently, theme words were those words, 4 letters and longer, that appeared the most times in the document. In case of a tie, the longer word was chosen.

  Another implementation detail Kupiec et al. do not discuss is how many theme words were selected, and how many ranked sentences were scored as "high" for this feature. Experiments showed that choosing the 10 most frequent words, and scoring the 10 most highly ranked sentences for this feature worked best. I examined the list of relevant words in each document, and found that generally, the top ten words reflected the theme of the document, and beyond this, there were many extraneous, unimportant words.

  Similarly, by looking at the scores of ranked sentences, I found that the highly ranked sentences tended to be among the top ten. After this, there were many tied scores, which indicated that past this point, looking at the frequency of theme words was less useful.

- **Uppercase Word Feature:** As noted in the Kupiec et al. paper, this feature was not useful for scientific journal articles. I doubted it would be useful in web documents, where formal terms are both less frequent and less significant than in scientific journals. However, this feature was included for the robustness. The measure was modified, however, so that an upper case word was considered relevant only if it belonged to the set of theme words. I also followed Kupiec et al.'s scoring convention of giving twice the score if the upper case theme word started the sentence. As with the theme word feature, the top 10 sentences were marked for this feature, as this value seemed to work best.

  In one of the data sets (dave data), the author chose to not capitalize any words at all. This is not entirely unusual with personal web pages. It does, however, render the uppercase word feature irrelevant. So for the dave data, this feature was not used.

- **Other features:** I examined and attempted a large number of other possible features. This included some of the features categories listed by (Paice 1990) and mentioned in (Kupiec et al. 1995). This included the title-keyword heuristic, other location heuristics, and other word cue heuristics. However, none of these gave useful results. Using more finely grained location information was no more helpful than using the feature described above. The title-keyword heuristic actually tended to degrade performance. The reason for this is that the web pages I looked at tended to have short titles that did accurately reflect content. For example, if the article was on politics, the title included "Salon: Politics – Gore at home?" The combination of short

titles with irrelevant words made the title feature useless. The articles I looked at also lacked major section headings, so it was not possible to exploit this.

Identifying additional features that served as good indicators for summary sentences would have been extremely valuable. Unfortunately, the many other possible features I examined were not useful with the training data I collected, and it was not easy to detect other possible features from the training data. So the system was limited to the features listed above.

Classifier

As noted, the Kupiec system is essentially a Naive Bayes classifier. At first, I used a strict Naive Bayes approach, using log probabilities to prevent underflow:

Choose s' such that $s' = \text{argmax}_{sk}[\log P(s) + \sum \log P(F_j \mid s)]$

where s means the sentence either is or is not a summary sentence. $P(F_j \mid s)$ is the probability of seeing the feature given that the sentence is (or is not) a summary sentence. Each of these were estimated using Maximum-Likelihood estimation. $P(s) = C(s) / C(w)$, where C(s) is the number of occurrences of summary sentences C(w) is the total number of sentences. $P(F_j \mid s)$ were estimated by

$P(F_j \mid s) = C(F_j,s) / C(s)$

where $C(F_j, s)$ is the number of occurrences of the feature F_j in sentence s, and C(s) is the number of summary sentences s in the training data.

Following a strict Naive Bayes approach, each sentence is classified as being a summary sentence if $P(s \in S \mid F_j)$ is greater than $P(s \notin S \mid F_j)$. In other words, it is chosen if it is more likely that it is a summary sentence than it is not. The results of this approach are shown in the results section.

The Kupiec system does not follow this approach. Instead of classifying each sentence, it uses the equation above to give a score to each sentence, which is the probability that the sentence is a summary sentence. It then chooses the sentences with the highest scores as summary sentences. It is not indicated in the paper how many of the top sentences are chosen, or how this number is decided.

I decided to try a similar approach as well: sentences are ranked using the naive Bayes scoring method. Then the n sentences with highest scores are selected as summary sentences. n is determined from the training data. If C(s) is the number of summary sentences in the training data, C(w) is the total number of sentences in the training data, and C(t) is the total number of sentences in the test document, then

$n = C(t) * C(s) / C(w)$

That is, it represents the number of summary sentences one would expect to find in the test document. Results using this approach are also found in the results section.

Evaluation Metric

I found the evaluation metric of Kupiec et al. surprisingly unclear. It is limited to the number of human summary sentences that are successfully returned by the system. There is no notion of false positives, so in their metric, a system could achieve 100% accuracy by including every sentence in the summary. I am certain that their system did not allow this, however, by limiting the number of summary sentences that could be chosen. However, this is never explicitly stated.

I evaluated my system based on both precision and accuracy. As previously noted, a cross-validation testing strategy was used, so the results are cumulative over every document in the training set. I used

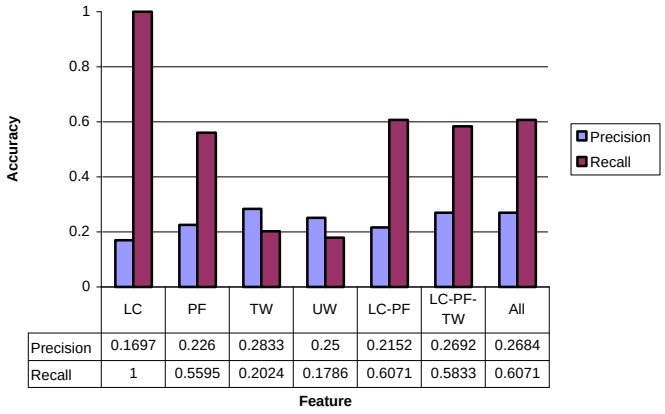
the same baseline performance metric as Kupiec et al: the performance of the system using only the sentence cut off feature, so that the system chooses the first sentences longer than the threshold length.

More specific implementation notes can be found in the submitted files (particularly summarizer.c). It is worth a quick lookover, if for no other reason the fact that an inordinate amount of time was spent on it.

Results

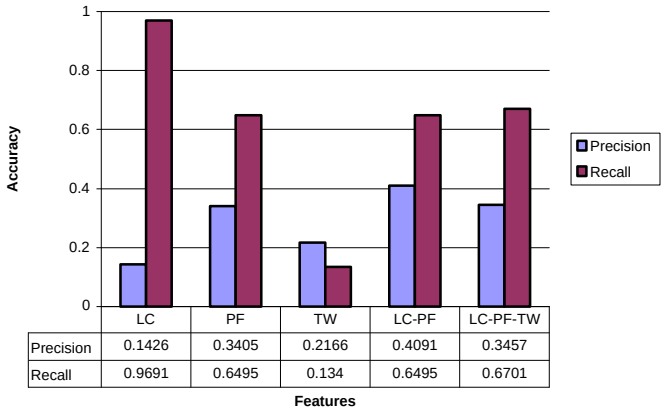
At first I attempted to use all of the training data (both salon and dave together) in training. This gave terrible results – the domains were just too different. Thus, in subsequent testing, each training set was tested separately. Much testing was done with various feature sets and different values for feature set thresholds, and other parameters. The results presented here represent the major results found after deciding on the best set of parameters.

The following are the results using a strict classifier, with no limit to the number of summary sentences chosen. In this system, a sentence is chosen as a summary sentence if the estimated $P(s \in S \mid F)$ (where s is the sentence, S is the set of summary sentences, and F is the set of features for s, is greater than $P(s \notin S \mid F)$. The system goes through and marks each sentence as a summary sentence or not based on this classification. This yields the following (the capital letters indicate the types of features used: LC – sentence length cutoff, PF – paragraph feature, TW – theme word, UW – upper case word)



	LC	PF	TW	UW	LC-PF	LC-PF-TW	All
Precision	0.1697	0.226	0.2833	0.25	0.2152	0.2692	0.2684
Recall	1	0.5595	0.2024	0.1786	0.6071	0.5833	0.6071

Figure 1: Performance of strict classifier on salon data



	LC	PF	TW	LC-PF	LC-PF-TW
Precision	0.1426	0.3405	0.2166	0.4091	0.3457
Recall	0.9691	0.6495	0.134	0.6495	0.6701

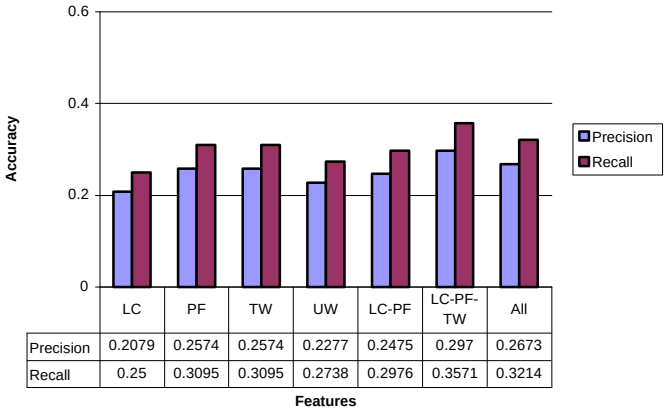
Figure 2: Performance of strict classifier on dave data

This method had terrible precision, and achieved high recall rates only by selecting nearly every sentence as a summary sentence. Because of the terrible disparity between recall and precision, it is difficult to see which feature set gives the best performance. It appears, however, that some combination of sentence length cutoff, paragraph, and theme word features performs best.

This is obviously a terrible summarization system, since it hardly does any summarizing. So the problem with using a strict classifier is that too many sentences are classified as being summary sentences, and the final results give very long summaries, which defeats the purpose of summarization.

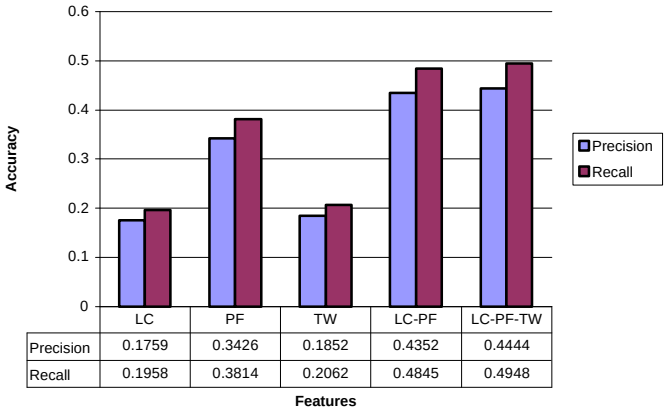
Thus, it was necessary to force the system to limit the summaries. So the next approach was to, instead of using a strict classification system, simply calculate the values of $P(s \in S \mid F)$ for each sentence in the test document and use this as a score for each sentence. The system then selects the highest scoring sentences as summary sentences. The number of sentences selected is determined by looking at the ratio of summary sentences to total sentences in the training data. Multiplying this by the number of sentences in the test data gives the number of summary sentences to choose.

This method yielded the following results (note that the scale is different from the first set of figures)



	LC	PF	TW	UW	LC-PF	LC-PF-TW	All
Precision	0.2079	0.2574	0.2574	0.2277	0.2475	0.297	0.2673
Recall	0.25	0.3095	0.3095	0.2738	0.2976	0.3571	0.3214

Figure 3: Performance of limited summary system on salon data



	LC	PF	TW	LC-PF	LC-PF-TW
Precision	0.1759	0.3426	0.1852	0.4352	0.4444
Recall	0.1958	0.3814	0.2062	0.4845	0.4948

Figure 4: Performance of limited summary system on dave data

This system kept a reasonable number of summary sentences. As expected, the system did better than the base line (LC) in both precision and recall. It appears the most useful features by themselves are the paragraph feature, the theme word feature, the upper case word feature, and then line count. The best combination is using the line count – paragraph feature – theme word. Adding the upper case word feature degrades system performance.

These results are in agreement with those of Kupiec et al. Their order of most useful features generally matches this one (although length cut-off is higher in their ordering), and the best combinations they found are similar to this system. The main difference is that this system found the theme word feature more useful than in the Kupiec system. In both systems, the upper case word feature degraded performance.

The precision and recall rates are disappointingly low, especially for the salon data. In both systems, less than half of all summary sentences were accurately chosen, and more than half of all chosen sentences were incorrectly chosen. While the best feature set does much better than baseline, it does not come close to human level performance. However, Kupiec et al. found similar results. On their corpus, their highest level of accuracy was 44% (for summary sentence recall). Performance on the salon data came short of this, but the system did better than this on the dave data. Although the numbers are low, they are consistent with established results on other domains, and the system does considerably better than a baseline system.

Examination of the results showed that the system chose sentences at the beginnings of paragraphs, if they were long enough. The theme feature distinguished between these, but the system never chose a short sentence, or one that did not begin a sentence. The vast majority of the missed sentences were those summary sentences that lay in the middle of a sentence.

These results were all achieved without performing any probability smoothing. It was expected that smoothing would not be as important as in other problem domains, since there are so few categories for each feature. Nevertheless, I looked at performing Lidstone smoothing to see if this would affect the results.

Predictably, Lidstone smoothing did not affect the system at all, unless the ë constant was set to a very high value (0.5 or above). This is expected, for reasons listed, and because sentences are chosen based on a ranking scheme. Smoothing would be unlikely to affect the scores of the most highly ranked sentences a great deal, especially if the amount of smoothing was very low. Thus, the set of sentences selected for summary would likely remain unchanged with smoothing.

So only high ë values affected the system at all. When smoothing with ë = 0.5 was used by the system, performance decreased or increased very, very slightly, depending on the feature set used. The differences in accuracy were so slight they are not presented here. The only notable affect was improving the performance of the best summarizer on the salon data, with a LC-PF-TC combination, to 0.3069 precision and 0.369 accuracy).

Discussion

As noted, the results obtained are similar to those of Kupiec et al. This was a somewhat surprising result, since the corpus used in that study (scientific journal articles) had much more structure and clear features, such as section headings, to guide sentence selection. Indeed, the fixed phrase feature they used was found to be the second best performing feature in sentence selection. Because of the disparate and unstructured format of the web documents examined here, such feature analysis could not be used. Thus, only simple features were examined, such as sentence length and position within a paragraph. Nevertheless, the results were similar to results of studies in other domains.

However, performance of the system was quite poor. It could not achieve even 50% accuracy in selecting sentences for summarization. Thus, the usefulness of such a system in automatic document summarization in web applications is dubious. This is especially true since the documents found on the web are so different. When the system was used on combined training data, it performed very poorly. It was necessary to separate the data by domain; the sites from which they were obtained. While the documents were consistent within the site, they were very different across sites, and the system did not perform well unless the domains were considered separately.

The best system tended to choose sentences that began paragraphs, pruning out short sentences, and using theme features to rank the paragraph headers. The system failed because many summary sentences are not paragraph headers. I found that different authors used different writing styles, and did not consistently place summary-like sentences as paragraph headers. In addition, because writers had different writing styles, I could not find consistent features that marked summary sentences. In short, it is no surprise that the system did not do well. The domain, web documents, is an extremely different one. Since there is so much variety and in many cases, lack of structure, it is difficult to employ statistical techniques with great success.

Because of this, there are several ways in which the system could be augmented that I did not have time to explore. While the scoring had a strong statistical foundation, it might be possible to use a different weighing scheme to get better results, by giving weight to more important features. I examined using variable weights a bit, but I could not find a principled way of adjusting these weights automatically, while keeping the scope of the project reasonable. This is one possible avenue of research.

Because there is so much variety in web documents, I believe that for summarization systems to work better, more linguistic information needs to be used. The Kupiec et al. system used certain heuristics, such as keywords, to track simple linguistic information, but to be generally applicable to web documents, using more detailed linguistic information is not just useful, but is likely necessary for good summarization

Finally, as noted, to get reasonable results, I had to separate the training data by domain. Another avenue of research to make a summarization system for web searches would be the automatic classification of documents by domains. Classifying web documents by domain would allow the system to be trained just within a domain, making it more useful than training it and using it with many disparate documents.

It should be noted that summarization is a hard problem even for humans. The Kupiec et al. study notes that the overlap in manually tagged documents is very low (25%), and even the same person changes their tagging of sentences over time. So while the accuracy of the system is low, we may not be able to hope for much more; even humans cannot agree on summary sentences. In light of this, and because their system performed much better than a baseline system, the Kupiec et al. paper considered their system successful.

This project can be considered successful by the same measure. It found that, contrary to expectation, the techniques used by Kupiec et al. can be applied to the much more difficult domain of web documents. The results achieved are very similar to that achieved by Kupiec et al, and in some respects better. Also, a system that used several features in scoring performed much better than a baseline system. The only disappointment was in not being able to extend the system successfully to get better performance. However, the final system did do relatively well.

There were several major lessons learned from this project:

- Most websites organize web documents so summaries are included in the beginning. Since these are so common, simple, dumb summarization techniques, such as the ones employed by search engines, do adequately well.

- Manually constructing text corpuses is extremely time intensive.
- A good system, one that uses a nontrivial selection algorithm, requires good heuristics and features. It is easier to find such features in certain domains than others. Thus, it is unlikely that a system will be able to find a good feature set that works well on all web documents.
- A good, more widely applicable system will probably have to use linguistic information. The lack of such information seems to be the primary weakness of this system.
- However, even with simple techniques, the system does a fair job at summarization; much better than a baseline system, and possibly not overwhelmingly much worse than a human. Thus, the techniques used by Kupiec are applicable to web documents as well.
- Some problems in NLP are hard. NLP-hard.