CS224N Project: Automatic Comment Summarization

Michael Levin, Suman Chakravartula, Adam Bliss

June 3, 2009

Introduction

It is very common today for websites to allow readers to comment on articles posted on the Internet. These comment threads usually contain interesting opinions and give a good indication of the average reader's sentiment. However, readers often post off-topic comments or post without reading all of the existing comments. This means that comments can be redundant or uninformative and that the sheer quantity of comments will quickly grow to an unmanageable size.

To address these issues, we developed comment set summarization systems. A comment summarizer takes a comment set and automatically produces a summary that is both indicative of the discussion and brief. We experimented with the following summarization approaches:

1. Baseline summarizers: randomly selected sentences, first few sentences
2. Sentence pruning using a Machine Learning model
3. Non-linear text-tiling with topic clustering

To collect the necessary data we downloaded articles and their corresponding comments from various websites (see Figure 1) and wrote a scraper script to automatically extract the article body, title, and author, along with the content and author of each comment. The script uses regular expressions to match the HTML code around each of the desired text areas. Although this approach is brittle and sensitive to changes in the website layout in the long run, it worked well for our purposes.

The summarization of a comment set is clearly not a well-defined task and the same set of comments would likely be manually summarized differently by different people. In light of this and to speed up the summarization process we defined a summary as a set of entire sentences extracted from the comments. A good summary should be no longer than 1500 words and only retain comments relevant to the article that offer new information. See Figure 2 for a birds-eye view of a summary.

To evaluate the quality of automatic summaries we used the ROUGE-2 score [3]. The ROUGE-n score compares n-gram co-occurrences between the automatically generated summary and the manually summarized "gold" summary. There are actually many variants of the ROUGE score, including different n-gram sizes. We selected ROUGE-2 because it had one of the highest correlations with human summarization evaluations while also being relatively simple to implement. The score is defined mathematically as

$$\text{ROUGE}_n = \frac{1}{N} \sum_{i=1}^{N} \frac{\min(\text{Count}_{\text{gold}}(n\text{-gram}_i), \text{Count}_{\text{test}}(n\text{-gram}_i))}{\text{Count}_{\text{gold}}(n\text{-gram}_i)}$$

where N is the number of different n-grams, $\text{Count}_{\text{gold}}(n\text{-gram}_i)$ is the count of n-gram i in the gold summary and analogously $\text{Count}_{\text{test}}(n\text{-gram}_i)$ is the n-gram count in the test summary.

If the test summary is longer or shorter than the gold summary, the ROUGE score will be affected even if the quality of the summary has not changed. To counteract this effect we specify the desired summary length in words that a summarizer should produce and set this value to the length of the gold summary that will be used to evaluate the automatically produced summary.

Baseline Summarizers

To establish a baseline for summarizer scores we first implemented a random summarizer that evenly sampled sentences from the comments until the desired summary length was reached. Because the desired summary length can be as little as 5% of the comment sentences, this summarizer is unlikely to hit sentences that are actually in the gold summary. Nevertheless because the ROUGE-2 score is comparing bigrams, there is still some chance of hitting a similar sentence. Also, frequenctly quoted sentences appear multiple times in the comment set and have an inflated chance of being selected, possibly more than once. Because this summarizer is randomized and has such high variance from run to run, scores are averaged over 50 runs.

Article			*Comment*	
Source	*Title*	*Words*	*Count*	*Words*
Slashdot	"High-Tech Start-Ups Put Down Roots In New Soil"	59	72	3,593
Slashdot	"Dot-Communism Is Already Here"	101	177	13,848
Slashdot	"How IBM Plans To Win Jeopardy!"	100	119	5,521
Slashdot	"Phony TCP Retransmissions Can Hide Secret Messages"	110	224	8,383
Slashdot	"An Argument For Leaving DNS Control In US Hands"	160	47	13,104
Slashdot	"Microsoft Rebrands Live Search As Bing"	100	107	3,283
Slashdot	"Evidence For Liquid Water On a Frozen Early Mars"	54	57	3,364
Slashdot	"Understanding Addiction-Based Game Design"	127	122	15,493
Slashdot	"Data Breach Exposes RAF Staff To Blackmail"	106	107	5,683
Slashdot	"Church of Scientology On Trial In France"	110	145	8,956
Slashdot	"Judgement Against Microsoft Declares XML ..."	101	132	5,333
Slashdot	"Palm Pre To Sync Seamlessly With iTunes"	104	137	6,841
Slashdot	"ASUS Designs Monster Dual-GTX285 4GB ..."	100	130	5,075
CNN	"Wikipedia bans Church of Scientology"	62	379	3,155
CNN	"Astronauts enjoy recycled urine"	149	306	5,585
CNN	"Microsoft's Zune HD to debut this fall"	78	418	6,538
CNN	"The strange concept of white holes"	129	264	12,819
CNN	"RFID freaks me right out"	77	161	7,269
CNN	"Obama: GM can recover with help"	110	214	6,735
CNN	"Are you a Facebook friend padder?"	132	291	8,795
CNN	"The death of BitTorrent?"	154	165	14,465
CNN	"California regulators rile ethanol producers"	73	432	7,478
CNN	"The battle over cybersecurity"	150	278	13,046
LinuxJournal	"What would you exchange Exchange for?"	71	170	6,882
LinuxJournal	"Why Microsoft Wants Us to Get All Mixed Up"	45	910	2,546
LinuxJournal	"The move to Linux, stymied by hardware"	50	466	6,574
LinuxJournal	"Nobody Uses Linux is Not a Good Enough Answer"	62	585	7,515
LinuxJournal	"Should Software Developers Be Liable for their Code?"	50	673	10,727
LinuxJournal	"Stallman vs. Clouds"	51	1,046	7,172
LinuxJournal	"Where are the Enterprise Management Tools ..."	50	1,156	7,272
LinuxJournal	"Who Owns Commercial Open Source ..."	40	1,001	2,983
LinuxJournal	"Should We Trash Windows Vista – or BadVista?"	43	971	3,041

Figure 1: *We collected data from the 33 articles listed above. Article gold summaries were written by different authors to ensure we did not overfit to a particular summarization style. The combined corpus has a total of 3,010 comments, 11,715 article and 245,230 comment words.*

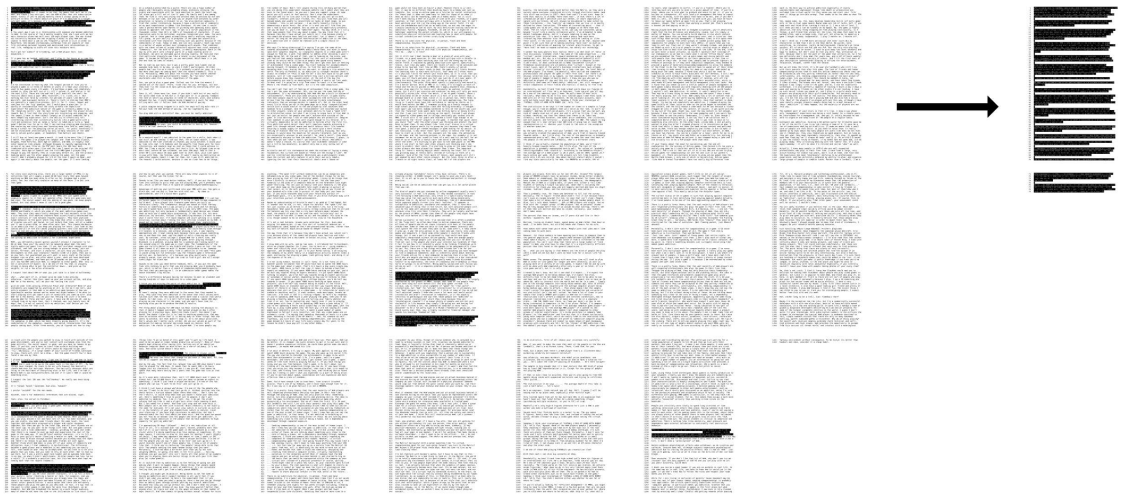


Figure 2: *Birds-eye view of the Slashdot article "Understanding Addiction-Based Game Design." Sentences selected for summarization are highlighted. The summary reduces a 15,943 word discussion to 719 words (less than 5% of the original size). The samples of automatically-generated summaries that we present are taken from this comment set.*

Random Summarizer				
Corpus:	Slashdot	CNN	LinuxJournal	Combined
Average ROUGE-2:	25.28%	27.04%	15.87%	23.26%

If we assume that the average reader is only willing to read a fixed number of sentences before giving up, we can establish another kind of baseline summarizer by simply choosing the sentences from the start of the comment set until the desired summary word length is reached. This summarizer more accurately reflects real-world comment viewing and also gains some accuracy due to the fact that the first few comments are less likely to be redundant and often set the direction of the following discussion.

Head Summarizer				
Corpus:	Slashdot	CNN	LinuxJournal	Combined
Average ROUGE-2:	30.89%	40.40%	22.96%	31.61%

Qualitatively, these summarizers generally do not choose very representative sentences. The returned summaries often have little coherence, are poorly related to the article, and in many cases contain badly-formed sentences. The following is a sample summary generated from the Slashdot article "Understanding Addiction-Based Game Design":

- `the addictive nature of the game is certainly added to by the dangling carrot and the next little reward that the player drives toward`
- `see giving up addiction to diablo 2 for a personal experience`
- `before wow it was backyardchickens`
- `and also whoosh`
- `i find that rather interesting because i base a definition of community on my past experiences in games like everquest`
- `you play wow with a controller`
- `but i can smoke for a month`
- `to what end`

If a game has me hooked, addicted, and I play it for hours at a time for weeks on end- fine. I'm getting enjoyment, the developers get money, everybody wins.

But it seems to me that the games that pull me in the most are those I buy outright, not the WOW-alikes that are subscription based. Surely if you're paying monthly there's always going to be a pressure on Devs to create addictive play? If I'm addicted to a bought-outright game, it's because it's a good game . That can't always be said for pay monthly games- the grind, the acheivements, the high-level horsie you just have to own- do they really add to the game, or do they just feed your addiction?

A few years ago I was in a relationship with someone who became addicted to WoW. In the course of the 8 months roughly that she lived with me her main character alone had well over 150 days played time. She was kicked out University because she was skipping classes to play WoW and was unable to hold down a job as she frequently called in sick in order to Raid/Farm. She became increasingly withdrawn from all aspects of real life including personal hygiene and maintained zero relationships in real life, managing to scare off even this hardcore nerd.

Still got her number? (I'm kidding, not a WoW player here. Just desperate.)

If a game has me hooked, addicted, and I play it for hours at a time for weeks on end- fine. You could say the same thing about drugs. Actually, you could say the same thing about drugs and I'd agree. If you're a user, or even an addict, and happy, that's fantastic. But that's hardly proof that addiction isn't insidious.

Both addictive drugs and addictive games can severely disrupt your life. However, once you become addicted to a drug, you're addicted for life. The addiction never entirely goes away. Addictive games are not like that at all, which is partly why I object to the use of the word "addicting" in this manner.

Figure 3: *The first few comments of the Slashdot article "Understanding Addiction-Based Game Design."*

- ◇ If a game has me hooked, addicted, and I play it for hours at a time for weeks on end- fine.
- ◇ I'm getting enjoyment, the developers get money, everybody wins.
- ◇ That can't always be said for pay monthly games- the grind, the acheivements, the high-level horsie you just have to own- do they really add to the game, or do they just feed your addiction?
- ◇ You could say the same thing about drugs.
- ◇ Actually, you could say the same thing about drugs and I'd agree.
- ◇ If you're a user, or even an addict, and happy, that's fantastic.
- ◇ Both addictive drugs and addictive games can severely disrupt your life.
- ◇ However, once you become addicted to a drug, you're addicted for life.

Figure 4: *First few sentences of the gold summary written for the comment set above.*

Machine Learning Summarizer

The core idea behind our Machine Learning summarizer is to train a classifier to detect whether a sentence was likely to be used in a summary based on a broad set of features. We use the classifier to produce the confidence that a sentence will be pruned for each sentence in the comment set, sort by this confidence, and build a summary of the desired length using the sentences predicted as most likely to be retained in the gold summary.

Choosing a classifier to use was not as simple as running various algorithms over the training data and using the one that had the highest classification rate. Since the learning summarizer only returns a fixed number of words with the lowest pruning scores, the ordering of sentences is more important than their classification. We experimented with various algorithms in Weka [5] including Naive Bayes, Random Forest, and Bagging classifiers but had roughly similar results with all of these. We decided to go with logistic regression because the algorithm is fast and simple. The algorithm also produces easily interpretable feature vector component weights that indicate the direction of the effect a feature has on the classification.

We experimented with a large variety of features (many inspired by [6]) some improved the results and some caused overfitting. Our final feature set consisted of the following features:

Positional Features

The older sentences near the start of the comment set tend to set the direction of the discussion, are less likely to be redundant, and are more important. Similarly, sentences near the beginning of a comment tend to set the topic for the rest of the comment. We expressed these intuitions with positional features:

1. Normalized comment position in document
2. Normalized sentence position in comment
3. Normalized sentence position of all sentences

Length Features

Sentences or comments that are too short are unlikely to have interesting content:

1. Comment length in words
2. Sentence length in words

Sentence Stems

We processed sentences in both the article and the comments to remove stop words and to stem words. Sentences with stems that appeared in the article body or title are probably more relevant and should be retained. We had several features that used this information:

1. Number of stems in the sentence
2. Normalized number of stems shared with article title
3. Normalized number of stems shared with article body text
4. Frequency of the most commonly occurring stem in sentence

Quoted Sentences

Comment threads commonly quote other comments. Comments that contain many quotes may be part of a discussion thread that has already been summarized. We tracked the first occurrence of each unique sentence and based several features on this information:

1. Sentence occurrence frequency in document if first occurrence
2. Sentence occurrence frequency in document if not first occurrence
3. Number of quoted sentences in comment

We also experimented with the following features but found that they tended to cause overfitting to the training dataset and hurt results during evaluation:

- Building a Naive Bayes classifier directly over words in the sentence to try to build a dictionary of words that are more or less likely to cause pruning and using this as an input signal
- Frequency of ad hominem or other specific words in the sentence
- Various features based on the starting or ending characters of a sentence

Because we have a limited number of articles, we use leave-one-out cross-validation to compute test results. For each article in the corpus we train the model on the other articles and test on the left-out article. The resulting scores are averaged together to produce the average ROUGE-2 score for the corpus.

Machine Learning Summarizer				
Corpus:	Slashdot	CNN	LinuxJournal	Combined
Average ROUGE-2:	43.98%	41.31%	23.77%	36.76%

The sentences produced by the Machine Learning summarizer tend to be longer and more relevant to the article. Because of comment-wide features and local similarity, sentences that appear in the same comment have a higher tendency to be retained or discarded together. The following is a sample summary produced by the Machine Learning summarizer:

◇ `if a game has me hooked addicted and i play it for hours at a time for weeks on end fine`
◇ `but it seems to me that the games that pull me in the most are those i buy outright not the wow alikes that are subscription based`
◇ `if im addicted to a bought outright game its because its a good game`
◇ `a few years ago i was in a relationship with someone who became addicted to wow`
◇ `in the course of the 8 months roughly that she lived with me her main character alone had well over 150 days played time`
◇ `you could say the same thing about drugs`
◇ `if youre a user or even an addict and happy thats fantastic`
◇ `but thats hardly proof that addiction isnt insidious`

Text-Tiling Summarizer

In this method, sentences in all comments are grouped together and the resulting document is treated as a single article of text. A non-linear text tiling approach [4] is used to recognize topic boundaries within comments and cluster the comment sentences. Each cluster contains sentences that are most closely related; thus each cluster can be viewed as a grouping of sentences that are about a distinguishable topic of the article. To generate the summary, sentences from each cluster are extracted using various intuitive strategies.

Comments are disparate pieces of text, each of which is a few sentences of opinion from its author about either the article or earlier comments. Due to this disparate nature, an "article" of comments (that is, the concatenation of all comments) lacks the structure of a typical document. However, we assume that there exists some hidden topical structure and that, by recognizing the hidden topics, individual sentences in the article can be clustered into a pool representing a certain topic.

As a first step, the concatenated comments are tagged using the Stanford Part Of Speech tagger. Since we are trying to separate the comments into various topics, noun tags can be very useful. Even though authors voice various opinions, they tend to talk about common topics which can be recognized by the nouns of sentences. Thus, sentences are stripped down to nouns alone. We use this condensed representation of sentences as input to the text tiling and clustering algorithm.

The basic method of text tiling is presented by Marti A. Hearst [1]. In this approach, the document is scanned once from the beginning to the end and topic boundaries are recognized along the way. The document is split at these boundaries into segments called tiles. This approach works for documents like news articles with definitive structures. However it cannot be assumed that this structure exists for comments. Hence, we follow a non-linear approach presented by Carlos N. Silla Jr. et all [4]. This method aims to group semantically related sentences irrespective of their position in the comment set.

The semantic distance between a pair of sentences is obtained according to

$$\text{Distance}(S_1, S_2) = \frac{1}{n} \sum_{i=1}^{n} D(W_i, S_1, S_2)$$

where n is the number of words (nouns) in sentence S_1. For a given word W_i in sentence S_1, $D(W_i, S_1, S_2)$ is given by

$$D(W_i, S_1, S_2) = \min_{W_j \in S_2} \text{WordDistance}(W_i, W_j).$$

The WordDistance(W_i, W_j) function is the word distance calculated using Wordnet's Hypernym relation. Suppose W_i, W_j are two nouns and DCA is deepest common ancestor between them. Further suppose that HDistance(n_i, n_j) indicates the hypernym distance between n_i and n_j. The WordDistance(W_i, W_j) is given by

$$\text{WordDistance}(W_i, W_j) = \frac{\text{HDistance}(W_i, \text{DCA})}{\text{HDistance}(W_i, \text{root})} + \frac{\text{HDistance}(W_j, \text{DCA})}{\text{HDistance}(W_j, \text{root})}$$

where 'root' is the root of the wordnet.

For each pair of sentences S_1 and S_2, the value of Distance(S_1, S_2) is stored in a square matrix called the distance matrix. This matrix provides access to the semantic distance between any pair of sentences, which is used in clustering.

In the final stage of the algorithm sentences are clustered using their semantic distance as given by the distance matrix. The number of clusters is a tunable parameter of the algorithm. Suppose K clusters are chosen. Each of these clusters is initialized to contain a pair of sentences. K closest pairs of sentences are chosen for this purpose. For each of the remaining sentences, distance between them and every cluster is computed by treating all words (nouns) in a cluster as a single sentence representation. The sentence/cluster pair with least distance is chosen and the sentence is engulfed into the cluster. This method is repeated until every sentence becomes part of some cluster. By the end of this procedure, K clusters are obtained where each cluster contains sentences that are semantically closely related and thus encode a certain topic in the document.

A summary is then generated by extracting sentences from the clusters until the word length constraint is reached. We can employ various strategies to choose the required number of sentences from the generated clusters. The strategies are explained below:

Uniformly weighted cluster strategy

In this strategy, we disregard the size of individual clusters. We try to extract the same or close to the same number of words from each cluster to generate the summary.

Text-Tiling with Uniform Strategy				
Corpus:	Slashdot	CNN	LinuxJournal	Combined
Average ROUGE-2:	32.29%	38.98%	21.02%	31.24%

Weighted cluster Strategy

In this strategy, we extract number of words proportional to the number of words in the cluster. A cluster with many words indicates that many authors have expressed similar opinion about a certain topic. The assumption is that such a topic deserves to have more sentences in the generated summary. The summaries generated by this strategy contained redundant sentences which translates to a lower ROUGE score and a bad summary.

Text-Tiling with Uniform Strategy				
Corpus:	Slashdot	CNN	LinuxJournal	Combined
Average ROUGE-2:	30.59%	35.76%	21.02%	29.55%

Comments often include nouns that belong to a specific domain of knowledge. For example, nouns such as Zimbra, Kolab and Citadel appeared often in linux journal articles. These do not exist in the wordnet. When normalized distance between sentences containing these words is computed, the presence of a nonexistent noun leads to suboptimal distance between the two sentences. The heuristic we used for nouns like these is to give them the same hypernym distance as the deepest hypernym distance known for that document. Topics can be segregated more accurately if a proper wordnet for these words existed.

The following is a sample summary generated by the text-tiling algorithm. The algorithm tries to choose sentences that are discussing different topics:

- `you play wow with a controller`
- `still got her number`
- `this is a consequence of the games design to some degree`
- `its the combination of the gameplay rewards and social interactions that keep people coming back`
- `so you are saying that mmos are only addictive and can not be good games`
- `she was kicked out university because she was skipping classes to play`
- `wow and was unable to hold down a job as she frequently called in sick in order to raid/farm`
- `however once you become addicted to a drug youre addicted for life`

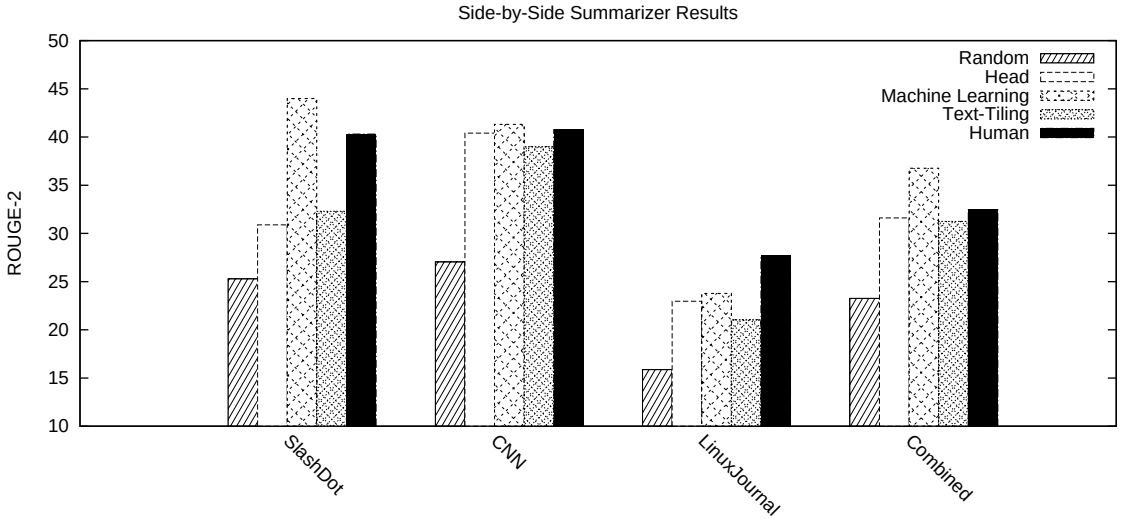
Future Work

One possible avenue for future work is to apply the techniques from online review mining. The goal of review mining is different since the goal is to produce a machine-readable output rather than a human-readable summary. Additionally the nature of the corpora are different because people posting comments on an article do not write the same way as people posting reviews of a product. However, review mining shares our emphasis on a set of disparate, brief comments. For example, in [2] Hu and Liu describe a technique for automatically identifying product features of the item under review. A similar approach could help our models identify distinct features of the article being commented on that commenters may be discussing.

For the text-tiling approach, we found that the summaries were very sensitive to the cluster size parameter arbitrarily chosen in the algorithm. Intuitively, the cluster size represents the number of different topics contained in the document. Since our data consists of comments about broad subjects like technology, politics and news, the number of inherent topics in documents varies significantly. For instance, in the article "Evidence For Liquid Water On a Frozen Early Mars" from the Slashdot corpus, when we choose the cluster size to be 45, we get a ROUGE score of 21.41%. However hand tuning the parameter to 65 yields a much better ROUGE score of 30.86%. Hand-tuning this parameter for every article is not feasible, therefore we tried to come up with an average that works for the most articles. We assumed that the number of topics in a document is directly proportional to its size and in fact we got the best results when the initial cluster size was set to 20% of the document size. We feel that a better way of choosing initial cluster size will lead to better results with this method. One possible approach would be develop document features and learn cluster sizes.

Conclusions

To give our results a reference frame, we manually resummarized a subset of nine of our articles and scored them in the same way we scored our automatic summarizers. The human-to-human summarization results are shown side-by-side with the automatic summarizer scores in the chart below:



It would seem that the effective range of the ROUGE score is actually fairly small at only 10-20% difference between random and human summarizations. Nevertheless we did observe a large qualitative difference in produced summaries with higher ROUGE scores. The ROUGE scores also seem to have a strong dependency on the source website, as can be seen with generally lower scores for LinuxJournal and higher for CNN articles.

Of the two NLP-based approaches we evaluated, the Machine Learning summarizer performed the best. For the Slashdot dataset the summarizer was even able to exceed the human-to-human performance level. This was most likely because the summarizer had tuned heavily toward the specific summarization styles used in the gold summaries for that corpus.

Overall, using natural language processing techniques we were able to develop effective comment set summarization algorithms using two different approaches. These algorithms have very useful applications for condensing comment sections on websites to an arbitrary size while retaining as many important sentences as possible.

Contribution

Michael Levin developed an extensible automatic summarization and testing framework, the Machine Learning summarizer, and wrote the report excluding sections concerning Text-Tiling. Suman Chakravartula developed the Text-Tiling summarizer and wrote related report sections. Adam Bliss assisted with features for the Machine Learning summarizer and report proof-reading. Article summaries were written by all three authors.

References

[1] Marti A. Hearst. Texttiling: segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, 1997.

[2] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA, 2004. ACM.

[3] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[4] Carlos N. Silla, Celso A. A. Kaestner, Alex A. Freitas, Rua Imaculada, and Conceio Curitiba Pr. A non-linear topic detection method for text summarization using wordnet. 2003.

[5] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[6] Klaus Zechner, Steve Finch, and Richard Shillcock. *Automatic Text Abstracting by Selecting Relevant Passages*. Masters dissertation, University of Edinburgh, United Kingdom, 1995.