

Update Summarization

Gaurav Aggarwal, Roshan Sumbaly, Shakti Sinha

June 5, 2009

1 Abstract

Update Summarization is a form of multi-document summarization wherein we generate a summary of a multi-document dataset based on the assumption that the user has already read a given set of documents. In our paper, we present a summarization system which clusters together sentences from the old set based on a semantic similarity score. We then use the centroids of these clusters, along with an information content score, to identify fresh or changed sentences in the subsequent set. These relevant sentences are ordered by their position in the original document and limited to 100 words to generate our update summaries. We discuss the components of our system, algorithms used and the motivation for choosing them. We also analyze the results at every stage and present our future work.

We used our system to generate update summaries for dataset available as part of the DUC 2007 Update Summarization task. ROUGE scores obtained for the summaries generated are comparable to other participants.

2 Introduction

With the advent of the internet we have seen a massive explosion in the amount of data one perceives in a day. In particular real time data seems to be the trend and is catching up. This ‘information overload’ can make it difficult for the user to consume new data as and when produced. The aim of our project is to build an ‘Update Summarization System’ which based on the user’s information need (either title or narrative) produces a summary of data which is either new or different.

A use case of this would be when a person has subscribed to news feeds and reads it every day. Ideally there would be multiple sources from which the news is being pulled resulting in similar articles being presented. Existing multi-document summarization

systems like [1] then remove redundant information. But still the user is faced with the problem of being presented with the information he would have read earlier. An update summarization system solves this problem by taking into account the articles read by user in the past during summarization thereby presenting only fresh or different information. Similarly another scenario wherein an update summarization can be helpful is in case of forums. A lot of replies on forums contains old ‘replied’ information. If we knew till which post the user has read in the past we could present a summary of only the new information found in the subsequent posts.

The common theme derived from the two use cases is that the input to our system would be some ‘past’ dataset, ‘present’ dataset and an ‘information need’ (implicitly the news article subject). The output would be a summary of the sentences in the ‘present’ dataset which would contain either new or fresh information with respect to the ‘past’ dataset. We built this system using the DUC 2007 Update Task dataset containing data from AQUAINT-2 collection. It contains news articles on 10 topics, each having three sets (A,B,C) which contain a total of 25 news articles. The 3 requirements are (a) normal summary of A (b) update summary of B, given that A is read by the user (c) update summary of C, given that A and B are read by user. In other words, for the second requirement the set A is our ‘past’ dataset, B is our ‘present’ dataset, while the topic of the set is the implicit query. Similarly for the third requirement, A and B are our ‘past’ datasets, while C is the ‘present’ set.

The first step of the implementation is to create a multidocument summary of A by clustering the sentences in the set. We judge whether a sentence of A belongs to a cluster or not on the basis of a *sentence similarity score* S , which we define later in section 4.3. After clustering, we take every cluster of A, and find one representative sentence having the maximum *information content* I , as described in section 4.5.1.

These sentences are then ordered according to their positions in the original documents and the summary of A is generated after applying the limitations (like ~100 words limitation for the DUC/TAC task).

The second step is to generate an update summary for documents in B. For this we use a *Novelty Detection* algorithm described in section 4.5. This algorithm takes every sentence of B and determines if it has new or different information. The sentences with high novelty are then run through the same process as A to generate the summary.

We can easily extend the above steps of update summary of B to that of C by running the check over all clusters formed using A and B. We only focus on document set B in this project. The above overview helps us break up the summarization step of set B into the following topics of interest :

- 1) Cluster Sentences to remove redundancy
- 2) Rank Sentences
- 3) Novelty detection
- 3) Generate Summary

3 Previous Work

Update Summarization was started as a Pilot Task in DUC 2007 [2] and later converted into a full fledged track in TAC 2008 [3].

One approach deals with statistical models built from the dataset. The work done by Maheedhar Kolla et. al. in [4] builds a ‘background model’ of the set A and then an updated probability model of a user selecting a word in set B, given the background model. They then score every sentence in B globally based on the probabilities of the terms and use it to rank and extract sentences until the summary is formed. A problem faced in their method is that if the sentence in B is completely filled with unknown words (wrt A), the smoothing part of the model takes over and this gives a constant score to it. This problem would not be faced in case of the DUC documents since they were hand-picked and have a similar language model, but would be faced in the real world scenario.

The authors of [5] weight a sentence globally using a Query Independent score (which is the prior probability of the sentence) and Query Dependent score. The Query Independent score is obtained after clustering at word level wherein terms in sets A and B having similar distribution are joined up together. The shortcoming of this approach is that by basing the score on just the distribution does not help in

capturing the semantic difference between sentences. This shortcoming is solved by [6] wherein the Query Independent score is based on the semantic similarity obtained using WordNet. They too generate a global score for every sentence in the set based on a linear combination of certain features like named entities, similarity with query / title, etc.

The work described above base their rankings on a global score and do no form of explicit clustering. This can result in some amount of redundancy of information [6]. This problem can be solved by comparing the chosen summary sentences with one another and trying to maximize the information content diversity between them thereby achieving the highest possible comprehensiveness. The work done in [7] does exactly the same by comparing pairs of sentences in A and B using 2 metrics (a) cosine similarity of the two sentence tf-idf vectors (b) linear combination of 3 factors viz. a unigram, bigram and syntactic factors. This approach once again relies only on word statistics not taking into account any form of semantic score.

Another approach has been to solve summarization as a global optimization problem with an integer linear programming solution [8]. Most of the work shown above aimed to concentrate on the ‘ranking’ aspect of the update summarizer. Use of Latent Semantic Analysis (LSA) to generate summaries [9][10] aims to ‘cluster’ together sentences.

The paper described above were all submitted to the DUC/TAC conferences. The above techniques aimed a lot towards the ranking and clustering aspects of the summarization. To learn more about the ‘generation’ aspect of summarization, literature study of the previous work done in the general area of multidocument summarization was done. Previous research on the ‘generate summaries’ part covers interesting ideas like sentence fusion and sentence compression.

[11] uses sentence fusion to generate summaries for news articles. They build up the first part of the system by clustering sentences into themes. Themes, as per their definition, are ‘groups of sentences from different documents that each say roughly the same thing’. The clustering of sentences is done on scores similar to the ones assigned in [6]. The novelty in this work comes in the latter half wherein sentence fusion is done by analyzing each sentence in a theme. A ‘basis tree’[11] is formed which contains information of the most commonly shared information in one theme. After pruning the basis tree a language model is used

to generate one sentence per theme.

Another aspect of generation is using sentence compression. [12] uses two techniques for the same viz. using a ‘noisy-channel’ model and ‘decision based conditional model.’ The inputs to these models is the parse tree of a sentence and the output is another parse tree of the compressed sentence.

4 Approach

In this section we describe the two main components of our system - summarization and novelty detection. We describe the motivation for algorithms used and implementation issues followed by error analysis.

4.1 Preprocessing

We starting by building a parser to extract relevant information from the corpus. The data was presented to us in the form of SGML [13], which is a markup language with a customizable syntax. A small snippet of one document has been shown in Figure 1. Since the paragraphs were tagged together, we required a boundary detection algorithm for the split to generate one sentence per line. We used Java’s BreakIterator class [14] which is provided in java.text package.

Before storing every sentence into a new file we also ran it through the Stanford POS tagger [15] and stored its tagged equivalent. For example the sentence *Jack and Jill went up the hill* is stored as *Jack#NNP and#CC Jill#NNP went#VBD up#RP the#DT hill#NN*. The POS tagged sentence would cut down execution time in later stages when calculating the word level similarity.

```
<DOC>
<DOCNO> AFW19991006.0166 </DOCNO>
<DATE_TIME> 1999-10-06 15:10:17 </DATE_TIME>
<BODY>
<CATEGORY> financial </CATEGORY>
<HEADLINE> Gov't Spent $13M on Microsoft Case </HEADLINE>
<TEXT>
<P>
    WASHINGTON (AP) -- The Justice Department's landmark antitrust
    trial and earlier legal actions against the Microsoft Corp. have
    cost $13.3 million over the past decade, the department said
    Wednesday in its first public accounting.
</P>
</TEXT>
</BODY>
</DOC>
```

Figure 1: SGML

4.2 Summarization of Set A

We use clustering to group sentences with similar meaning together. For this we require a metric called Sentence Semantic Score S which captures the distance between a pair of sentences. The summary is created by selecting one representative sentence from every cluster.

4.2.1 Word Pair Semantic Score

Sentence Semantic Score S depends on semantic similarity score between the corresponding words $\lambda(w_1, w_2)$ in the sentences being compared. To compute this similarity, we use WordNet to extract word-sense or concepts corresponding to the words and try to find a measure that captures the relationship between these concepts.

The literature talks about two measures : ‘measure of similarity’ (MOS) and ‘measure of relatedness’ (MOR) between pairs of concepts.[16] MOS is a metric which quantifies how two concepts are similar based on the *is-a* relation. For example, *apple* and *fruit* might be considered having a common ancestor when we draw a hierarchy of *is-a* relation. On the other hand, MOR is able to capture a wider set of relations (like *has-part*, *is-made-of*, *is-an-attribute-of*, etc) and hence captures more concept pairs. For example, pairs like *car* and *wheel* would be quantified well by MOR but not by MOS. Another advantage of MOR over MOS is that it works across different parts-of-speeches. We adopt MOR for our system since it is able to capture more ‘semantic’ meaning compared to MOS.

We can now broadly classify the different MOR metrics proposed in literature into 2 groups viz. edge counting-based methods and information theory-based methods.[17] The edge counting-based methods try to score the concept pairs by finding the shortest path in their intersecting relation graphs. The information theory method instead tries to give a score on the basis of the gloss associated with the two senses. We used ‘Extended Gloss Overlap (lesk)’ measure (ω) defined in [19].

Implementation : We make use of WordNet (version 2.1), a large lexical database which contains information regarding all the relations mentioned above. Further, we used a Perl package called WordNet::Similarity [16]. This package has 3 MOR metrics coded viz. *hso*[18], *lesk*[19] and *vector*[20]. We make use of the algorithm in [19], which uses the informa-

tion theory method. To invoke this Perl package in Java, we initially tried using the `Inline::Java` package [22], which is a Perl interpreter for Java. This method did not work. So we started invoking the Perl script as a different process in Java, with its output redirected to a file.

We next faced the problem that an invocation of a process for every concept pair would take up a lot of execution time. As an optimization step, we ran the script over all documents in set A, B and C once and dumped the serialized `CounterMap` object containing the concept pair similarity scores into a file. For future processing, it was only required to deserialize the dump once thereby cutting down on the execution time.

Analysis : Following are some of the concept-pairs and their corresponding scores. If we had adopted any of the MOS metrics, these values would have been zero.

w_1	w_2	Semantic Score ω
Union	membership	0.9090
European	France	0.8938
annual	monthly	0.5229
early	shortly	0.4066
future	today	0.4642

Table 1: Word pair semantic score examples

One problem at this stage is when certain words are not present in WordNet and hence the `Wordnet::Similarity` package returns an error. In such cases we use $l = \frac{1}{1+E}$ as the similarity score, where E is the Levenstein Edit Distance. But on experimenting we noticed that certain word pairs got a very high score compared to other pairs due to the dominance of edit distance. For example, word pairs like *Joe-Toe* gets a high score since they are not present in WordNet and their edit distance is low. To solve this problem we set the score as a linear combination of edit distance and the `WordNet::Similarity` score i.e. $\lambda(w_1, w_2) = \alpha\omega + (1 - \alpha)l$ where $0 \leq \alpha \leq 1$.

4.3 Sentence Similarity Score

Now that we have a word pair semantic score, we focus on finding semantic score between two sentences. This score should be able to capture the extent of semantic overlap for any given two sentences. The simplest way to do so would be to take a one-one alignment between the words in the 2 sentences and

then add them up. This would not be appropriate since most sentences having similar meaning do not have an one-to-one alignment. This is similar to the problem faced in MT systems.

4.3.1 Word Alignment

Implementation : We can formulate this problem of finding the correct alignment as an assignment problem wherein we're trying to find the maximum weight matching in a bipartite graph. A simple way to solve this matching problem is to use an greedy heuristic algorithm which tries to find for every word in one sentence the local best possible matching word in the other sentence. The algorithm for the same is given below. In the algorithm we have two sentences viz. $X[1..i..N_X]$ and $Y[1..j..N_Y]$ with their corresponding word semantic scores stored in matrix $WS[N_X][N_Y]$. Here $WS[i][j] = \lambda(w_i, w_j)$.

```

matchScore = 0
match[N_X] = null
for each word in X, X[i]
  bestCandidate = bestScore = -1;
  for each word in Y, Y[j]
    if Y[j] is still free && WS[i][j] > bestScore
      bestScore = WS[i][j]
      bestCandidate = j;

  if bestCandidate != -1
    match[i] = bestCandidate
    matchScore += bestScore

```

The output of this algorithm is the array ' $match[1..i..N_X]$ ' which contains the corresponding index j for every word i . It also returns the total score ' $matchScore$ '. This algorithm works well when we need a quick matching of word pairs. But on running this on our set, we noticed that it does not always give the optimum answer. In particular we encounter certain words which could have found higher matching corresponding words, but due to their late appearance in the sentence, miss out on the chance to do so.

Hence we next implemented a better assignment algorithm called Kuhn-Munkres algorithm [21] (originally called Hungarian algorithm). This algorithm takes as its input a complete bipartite graph with weighted edges and finds the assignment which would give a global minimum. Relating this to our case, our bipartite graph would have as its first set words from the first sentence while the second set having words from the second sentence (represented as an

adjacency matrix of dimension $N_X \times N_Y$). The edge weights would be $1 - sim(w_1, w_2)$, where w_1 is a word (node) from first set, w_2 is a word from the second set while sim is the word pair semantic score. The output would be the same i.e. $match$ and $matchScore$.

Analysis : Given below is an example of the matching obtained by Heuristic as well as Kuhn-Munkres algorithm for the sentences given below. It is evident that since the word ‘Madrid’ comes earlier on in the first sentence, it takes up the aligned word ‘European’ from the 2nd sentence. But this is wrong since it should have been assigned with the same word ‘European’ in the 2nd sentence. This error is rectified by Hungarian.

Sentence 1 - *At the Madrid summit last December, leaders of EU member nations agreed unanimously that the European single currency will be formally launched on January 1, 1999*

Sentence 2 - *Despite skepticism about the actual realization of a single European currency as scheduled on January 1, 1999, preparations for the design of the Euro note have already begun.*

w_1	w_2 Heuristic	w_2 Hungarian
Madrid	European	skepticism
summit	design	design
December	January	Despite
leaders	scheduled	scheduled
EU	Euro	Euro
member	begun	note
nations	single	preparations
unanimously	actual	realization
European	note	European
single	Despite	single
currency	realization	currency
formally	currency	begun
January	preparations	January
1	1	1
1999	1999	1999

Table 2: Alignments

4.3.2 Similarity Metric

Implementation : We define the Sentence Similarity Score S as:

$$S = \frac{2 \times matchScore}{|X| + |Y|}$$

$$matchScore = \sum_{\forall i,j:(match[i]=j)} WS[i][j]$$

This $matchScore$ is calculated according to the above formula and returned by the alignment algorithm. By multiplying the numerator by 2, our aim was to allow every individual similarity values to reflect their contributions to the overall score S , while the denominator normalizes by length.

Analysis : Table 3 and 4 present the scores received when comparing two similar sentences (Sentence 1 & 2) and two dissimilar sentences (Sentence 1 & 3). As is evident from the Similarity Score S for the similar sentences is very high (0.38) compared to the dissimilar sentences (0.039).

Sentence 1 - *At the Madrid summit last December, leaders of EU member nations agreed unanimously that the European single currency will be formally launched on January 1, 1999.*

Sentence 2 - *The European single currency euro will go ahead on schedule on January 1, 1999 with a broad membership, according to a survey of some prominent British economists.*

Sentence 3 - *The European Monetary Institute, responsible for the preparations, has set up a consultative group of artists, art historians and psychologists to come up with designs.*

4.4 Clustering

Our next requirement to generate summary of set A is to find sentences which are representatives of a set. This motivates us to cluster together sentences which have the same semantic meaning (using the measure S defined above) and find one sentence which would represent all of them. We adopted the simple k-means algorithm [23] over hierarchical clustering since (a) we required a representative sentence. The centroids formed in the case of k-means act as natural representatives. We used the sentence having the least minimum average distance from everyone else in the cluster as our centroid. (b) the requirement of our problem was to generate flat clusters. Our problem of similar sentences is slightly tough to model as an hierarchical clustering problem since finding the correct threshold of termination can be tough. Since our

w_1	w_2	Word Pair Score
Madrid	ahead	0.0166
summit	survey	0.0200
December	economists	0.0126
leaders	membership	0.1428
EU	euro	0.0333
member	prominent	0.0111
nations	British	0.0392
agreed	according	0.0297
unanimously	broad	0.0770
European	European	1.0000
single	single	1.0000
currency	currency	1.0000
formally	schedule	0.0411
January	January	1.0000
1	1	1.0000
1999	1999	1.0000
S	-	0.3893

Table 3: S for Sentence 1 & 2

w_3	w_1	Word Pair Score
European	European	0.1105
Monetary	January	0.0200
Institute	economists	0.0611
responsible	single	0.0325
preparations	currency	0.0118
set	broad	0.0560
consultative	ahead	0.0076
group	membership	0.0626
artists	schedule	0.0711
art	euro	0.0250
historians	British	0.0611
psychologists	prominent	0.0091
come	according	0.0200
designs	survey	0.0464
S	-	0.0396

Table 4: S for Sentence 3 & 1

requirements were more biased towards flat clusters we adopted k-means.

4.4.1 Approach 1

Our first approach was to use normal k-means clustering with random seed sentences as centroids and an arbitrary number of clusters k . This approach, after termination, results in centroids which don't have much semantic information. For example two of the centroids formed are 'We have to look forward and be prepared' and 'It is firstly and mainly a political

plan for us.' As is evident these centroids don't hold enough semantic meaning. We attribute this to two reasons (a) bad seed set (b) wrong number of clusters k .

4.4.2 Approach 2

To further refine the results of our previous approach we changed the number of clusters and set it to the number of documents. We also changed the initial centroid seed set to have the first sentence of every document. The intuition behind this was that the first sentence in a news article usually represents the gist of the article. We also added the headlines of the articles into the corresponding seed clusters.

This approach gives comparatively better results with the exception of certain sentences being misclassified. These misclassified sentences have the problem that the presence of certain words increases its similarity score, thereby resulting in them being misclassified. For example, we have a certain centroid 'The European Union member states are required to completely replace their own national currencies with the Euro from January 1, 2002' and one of the sentences which falls into its cluster is 'BRUSSELS, January 22 (Xinhua) Two recent Eurobarometer opinion polls showed that a majority of European Union(EU) citizens are in favor of the single currency'. Their alignments and word pair semantic scores are shown in Table 5. As is evident from the sentences their semantic meanings are totally different. But the S score obtained gives a high value of 0.28641. This problem is primarily due to certain word pairs like *European-European*, *Union-Union*, *currencies-currency* and *January-January* who receive a very high word pair semantic score of 1.0.

4.4.3 Approach 3

We observed that most of the words causing the above problem were commonly occurring words that are present in almost every sentence of the dataset. Hence the contribution of these commonly occurring words towards the S score should be less. To take this into account we associate a certain cluster centric weight with each word. We define this 'cluster centric word weight' $\tau(w_i)$ as:

$$\tau(w_i) = \frac{tf_{cluster}}{tf_{overall}} \times idf_{overall}$$

τ is a variant of tf-idf score. In this formula $tf_{cluster}$ is the term frequency of the word in the corresponding cluster, $tf_{overall}$ is the term frequency of the word in the A dataset and $idf_{overall}$ is the idf computed at sentence level. High values of $\frac{tf_{cluster}}{tf_{overall}}$ would mean that the word is more informative about that particular cluster and high values of $idf_{overall}$ means that the word is rare in the sense that it is not used in almost all sentences.

This weight is now accounted for in the similarity metric S :

$$S = \frac{\sum_{i,j:match[i]=j} \lambda(w_i, w_j) * (\tau(w_i) + \tau(w_j))}{\sum_{\forall i} \tau(w_i) + \sum_{\forall j} \tau(w_j)}$$

This change of similarity metric removes such outliers thereby resulting in an improvement of the clusters. With respect to our previous example in Table 5, value of weight τ for the words *European*, *currency*, *January* is very small compared to other words. Hence its contribution to the overall score decreases thereby giving a better overall S score of 0.056509 (vs 0.28641) for the two dissimilar sentences.

We face another problem after the above change in definition of S . In many of sentence comparisons, we observed that sum of weights τ for all words in one sentence s_1 was quite small as compared to the other sentence s_2 . This might be because s_2 was the centroid of a cluster and s_1 had little semantic overlap with it. So many of the words in s_1 would have zero weight because they are not informative about the corresponding cluster of which s_2 is the centroid. For example, consider the two dissimilar sentences $s_1 = \text{'Aung San Suu Kyi , the Burmese Nobel Peace laureate , may be the most spied-on , or surveilled , human being on earth'}$ and $s_2 = \text{'Nobel laureate Aung San Suu Kyi offered Thursday to end her road side standoff with the military government if the government agrees to release jailed members of her political party'}$. In this case, $\sum_{\forall w \in s_1} \tau(w) = 4.285$, $\sum_{\forall w \in s_2} \tau(w) = 29.689$ and the numerator equals 9.022. By the above formula, we get $S = 0.2655$ which is artificially high.

Hence we once again modify our sentence similarity score to:

$$S = \frac{\sum_{\forall w_j \in s_k: match[i]=j} \lambda(w_i, w_j) * (\tau(w_j))}{\sum_{\forall w_j \in s_k} \tau(w_j)}$$

$$k = \arg \max_{i,j} \left(\sum_{\forall w \in s_i} \tau(w), \sum_{\forall w \in s_j} \tau(w) \right)$$

With this new definition, S becomes 0.1595 which is a more realistic prediction of the score.

4.4.4 Approach 4

After applying the various tweaks mentioned above we still noticed certain sentences which should not have been present in their current clusters. These sentences had a totally different semantic meaning from not only the sentences in the cluster but also from centroids of other clusters. An example of this is the centroid *Exiled dissidents from Myanmar urged Thai Prime Minister Chuan Leekpai on Friday to reschedule a trip to Myanmar by his top general because it will coincide with the anniversary of the military government's takeover*. whose cluster has the off sentence *In 1988, the military gunned down thousands of protesters during a nationwide democracy uprising*. This sentence could not be classified into any other cluster and got into this one due to overlap with the less important words.

The solution to this problem is to allow variable number of clusters. Hence when a sentence is checked with all clusters and cannot be classified into any it is automatically made into a new cluster. At the end of every iteration we run the function which computes the new cluster centric τ weight of words and then continues the same process.

4.5 Novelty Detection

Once the summaries of the documents in set A of the corpus have been generated, the next task is to determine the update summaries of the documents in set B. The update summary of set B is the summary created assuming that the user has seen the documents in part A. This is a novelty detection task, where given the previous information of set A represented through its summary, we determine which sentences of set B are new. Our approach to novelty detection involves the use of a metric which we call Information Content, I .

Centroid(w_i)	Sentence(w_j)	$\lambda(w_i, w_j)$	$\tau(w_i)$	$\tau(w_j)$
European	European	1.0000	0.0658	0.0658
Union	Union	1.0000	0.2349	0.2349
member	BRUSSELS	0.0223	0.0000	1.1224
states	citizens	0.0841	0.0000	4.0604
required	recent	0.0195	0.0000	1.7748
completely	opinion	0.0232	0.0000	4.0604
replace	single	0.0401	0.0000	0.0951
national	majority	0.0336	0.0000	1.2571
currencies	currency	1.0000	0.0531	0.0531
Euro	EU	0.0333	0.0335	0.2835
January	January	1.0000	0.2460	0.2460
1	showed	0.0142	0.0000	4.0604
2002	22	0.0333	0.0000	2.0302

Table 5: Example showing the significance of the new weights τ

w_1	w_2	Word Pair Score	tf-idf(w_1)	tf-idf(w_2)
Nobel	Nobel	1.0000	0.87	0.87
laureate	laureate	1.0000	1.50	1.50
Aung	Aung	1.0000	0.59	0.59
San	San	1.0000	0.57	0.57
Suu	Suu	1.0000	0.38	0.38
Kyi	Kyi	1.0000	0.38	0.38
standoff	spied-on	0.0025	0.00	5.89
agrees	Burmese	0.0033	0.00	2.87
release	Peace	0.0149	0.00	1.66
jailed	surveilled	0.0028	0.00	5.89
political	human	0.1002	0.00	3.21
party	earth	0.0107	0.00	5.89

Table 6: Example showing the modified S metric

4.5.1 Information Content

The information content of a sentence is a value which gives us the amount of information present in the sentence, with respect to a given topic. For a sentence S , this score can be computed as

$$I = \sum_{w \in S} i_w$$

where i_w is the information content of the word w . The information content of a word w , in turn, is given as

$$i_w = IDF_w \times Context_w$$

The score $Context_w$ gives a value indicating how close the word w is to the words occurring in the topic. It is computed by taking the reciprocal of the minimum number of sentences through which we can connect word w to any topic word through cooccurrence. Also,

we define the information content of stop words to be zero.

4.5.2 Information Content Scores for Sentence Pair

For the task of update summarization, we need to compare the information content of two sentences. Specifically, we have to compute metrics such as which sentence has more information, how much common information is present in the sentences, etc. Let us define these metrics.

The common information for two sentences S_1 and S_2 is given as:

$$I_c = \sum_{w_1, w_2 | S_{w_1 w_2} > t} \frac{i_{w_1} + i_{w_2}}{2} \quad (1)$$

where w_1 and w_2 are the aligned words from the two

sentences, $S_{w_1 w_2}$ is the word similarity score for the two words and t is the threshold over which we consider words to be similar.

The residual information content of a sentence in a sentence pair is the difference between its information content and the common information content of the pair.

$$I_r = I - I_c \quad (2)$$

4.5.3 Computation of Update Summary

Our method for computing the update summary for the document set B, given the summary of document set A is given below. The summary of set A is represented by the sentences that constitute its cluster centroids computed during summarization.

1. Create a current list of centroids, and add all the centroids from set A to it.
2. For each sentence in set B, compute its information content scores I_c and I_r with respect to every centroid in the list.
3. If the common information I_c is less than the residual information I_r of both sentences, then create a new centroid for set B using the sentence from B being considered. We do this because there is no existing centroid which shares a significant amount of information with the sentence from B.
4. If the common information I_c is greater than at least one of the residual information, then we merge the sentence being considered into the cluster which has the highest common information content. If the residual information of the sentence being considered is greater than the residual information of the centroid, then we replace the centroid by this sentence.
5. We repeat steps 2-4 till all the sentences from set B have been considered.
6. We now compute the information density of each sentence in each new cluster created. The information density is the information content of the sentence divided by its length. We replace the centroid by the sentence with the highest information density. This ensures that the centroid is a sentence which has fewer stop words and more useful words.

The new centroids created on following the above steps give us the update summary for the set B.

5 Evaluations

To evaluate our system, we calculate the ROUGE[23] score between update summary produced by human judges and those generated by us. Besides, we also calculated ROUGE scores for other participants at the DUC 2007 conference. This allowed us to compare our system’s performance with others. We focus only on ROUGE-2 since ROUGE-4 scores were quite low for all participants. ROUGE-2 measures the fraction of bigrams in common between the human and machine summaries.

We evaluated our system on three topic datasets - ‘Introduction of Euro Currency’, ‘Burma Government Change’ and ‘Anti-trust lawsuit against Microsoft’. There are four human summaries available for every topic. We compared all machine summaries with each of the four human summaries and report the best ROUGE scores obtained by our system. Figure 2 plots the ROUGE scores and Table 5 shows rank of our system when compared with other participants for the three topics. Following shows the update summary generated by our system for the Microsoft topic.

1993: Feb. 5 _ On a 2-2 vote, FTC deadlocks on taking action against Microsoft. The agency ponders charges that Microsoft illegally thwarted competition while adding hidden codes to its operating system software in a bid to hinder competing applications vendors. ‘‘California is so impacted by the colossus of Microsoft, and so many businesses were impacted and felt that their competitive opportunities were smothered,’’ Lockyer said. Sept. 10 - Each side files another stack of voluminous rebuttal briefs that largely review their prior arguments while disparaging the opponents’ case.

These results show that our system’s performance (as measured by ROUGE) is comparable to the other participants. The update summaries produced contain only new information from documents in B dataset. However, sentences are not always selected in the correct order of their importance with respect to the overall information need. Also, consecutive sentences in the summary are not always linguistically connected to each other because we pick sentences from documents in B without checking for continuity.

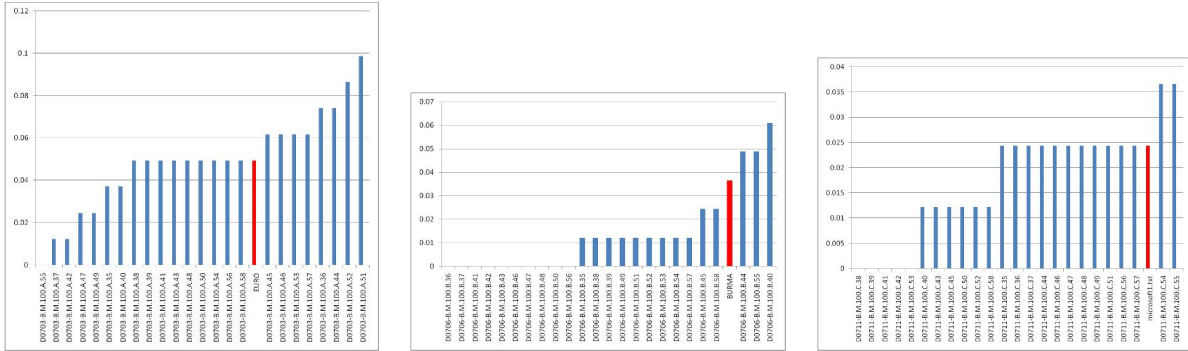


Figure 2: ROUGE-2 scores of human summaries when compared with human summaries. Our system is shown in red. Left: Euro, Middle: Burma, Right: Microsoft.

	BURMA	MICROSOFT	EURO
Gold 1	25	14	17
Gold 2	19	3	21
Gold 3	20	11	17
Gold 4	4	10	9

Table 7: Rank of our system out of 25 total machine summaries

6 Analysis and Future Work

Though our Sentence Similarity Score S tries to capture semantic relatedness using WordNet, it is still influenced mostly by surface similarities between words in the corresponding sentences. We observed a number of pathological problems related to this during our analysis.

Firstly, the process of word alignment ignores the context in which a word appears and tries to greedily optimize the overall similarity score by matching the word with the best possible word in the second sentence. For instance, when the sentence containing the fragment - *...calls to release political prisoners or begin a dialogue...* is aligned with sentence containing fragment - *...agrees to release jailed members of her political party...*, the word *political* in the first sentence is aligned to word *political* in the second sentence because their word level similarity score is 1. But this alignment is very improbable given the context in which *political* appears in the two sentences. These errors can lead to artificially high similarity scores and misguide the clustering process. We could rectify this by identifying phrases that frequently appear together and look for an alignment that tries to preserve these properties.

Dependencies among different parts of speech in a sentence could also be used to add more context

to the similarity computations. Verbs and adjectives usually have associated arguments. Two sentences could use highly related verbs or adjectives but could still mean completely different things because the arguments to which the verbs or adjective apply are very different. For example, the two sentences *...according to a statement released Thursday by ...* and *...if the government agrees to release jailed members...* contain the same verb *release* but it applies to different arguments *statement* and *jailed members* respectively. Thus the two sentences are semantically different but get comparatively high S score because of high word alignment score for *release*. Semantic role labeling using PropBank and FrameNet could be used to add such semantic information.

The sentence similarity score also performed poorly when a sentence contains references to entities or concepts in other sentences in the document. For example, one document talks about ‘a demand made by Suu Kyi to the government’ as evident in the fragment *...Suu Kyi offered Thursday to end her road side standoff with the military government if the government agrees to release jailed members...* Other sentences later in the document give more information about the response to this demand - *The military said Thursday it has no intention of meeting the demand*. But since our sentence similarity only works with words present in the sentence, it has no in-

formation about the word *demand* or *military* refers to and hence gives low similarity score to this sentence pair. Co-reference and Anaphora resolution tools could help in this regard.

KMeans clustering to group sentences with similar semantic information performs well when the news articles talk about semantically different topics and sentences within a document are ‘sufficiently’ similar to other sentences in the document. This was the case with the document set on ‘Introduction of the Euro currency’. However, articles in other document sets contain more historical information as mentioned in Section 4.4.4 which does not overlap significantly with any of the seed centroids. For every such sentence, we add a new cluster with this sentence as the centroid and continue redistributing remaining sentences by calculating similarity against all old and new centroids. This approach may be susceptible to the order in which we pick sentences during redistribution. Besides, it is difficult to identify the informative words (so that we could give them higher weights) for the newly added cluster as it contains only one sentence at that point. To solve this problem, we could try using hierarchical clustering for sentences that don’t overlap significantly with any centroid. Hierarchical clustering starts with every sentence in its own cluster and merges two most similar clusters iteratively. So we can avoid the initial seed selection problem and also the clustering output would not depend on the order of sentences. The resulting clusters could then be combined with clusters obtained using KMeans earlier.

During our analysis, we observed that most sentences contain few important words that contribute significantly to the overall meaning of the sentence. Humans are easily able to identify these words and segregate them from other contextual or historical information in the sentence. For example, in the sentence - *While Myanmar’s military government continued its arrests of the country’s democracy activists, a top general accused Nobel Peace Prize winner Aung San Suu Kyi of working with terrorists, state-run newspapers reported Thursday.*, important concepts include that ‘a top general’, ‘accuse’, ‘Suu Kyi’, ‘terrorists’. Other concepts including ‘arrests’, ‘activists’, ‘newspaper’, ‘report’ are not crucial to the overall meaning. Identifying such words would help us set appropriate weights for the similarity calculations and hence the resulting clusters should be more semantically coherent. Sentence Compression or paraphrasing techniques could be a step in this

direction.

7 Conclusion

In this project, we explored different techniques for generating multidocument summaries and update summaries. Our results on the data sets provided by DUC 2007 demonstrate that our algorithms work reasonably well, and perform comparable to the middle-tier teams. Through error analysis, we have explored improvements that can be made to our methods, which we believe can produce significantly better results.

8 Contribution

Gaurav worked on Word pair similarity, Roshan on Sentence level similarity while Shakti built the Information Content part. The clustering section was programmed in collaboration. The error analysis and documentation was equally split.

9 References

- [1] Columbia Newsblaster - <http://newsblaster.cs.columbia.edu/>
- [2] Document Understanding Conferences - <http://www-nlpir.nist.gov/projects/duc/index.html>
- [3] Text Analysis Conference - <http://www.nist.gov/tac/>
- [4] Comparison of models based on summaries or documents towards extraction of update summaries - Maheedhar Kolla et.al
- [5] IIIT Hyderabad at DUC 2007 - Prasad Pingali et.al.
- [6] A Semantic Summarization System: University of Birmingham at TAC 2008 - Abdullah Bawakid et. al
- [7] TAC 2008 Update Summarization Task of ICL - Sujian Li et. al
- [8] The ICSI Summarization System at TAC 2008 - Dan Gillick et. al.
- [9] Using Latent Semantic Analysis for Extractive Summarization - Kirill Kireyev et. al.
- [10] SUTLER: Update SummarizER based on Latent Topics - Josef Steinberger et. al.
- [11] Sentence Fusion for Multidocument News Summarization - Regina Barzilay et. al.
- [12] Summarization beyond sentence extraction: A probabilistic approach to sentence compression -

Kevin Knight et. al.

[13] Standard Generalized Markup Language - <http://en.wikipedia.org/wiki/SGML>

[14] Boundary Detection - <http://userguide.icu-project.org/boundaryanalysis>

[15] Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger - Toutanova et. al.

[16] WordNet::Similarity - Measuring the Relatedness of Concepts - Ted Pedersen et. al.

[17] Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures - Alexander Budanitsky et. al.

[18] Lexical chains as representations of context for the detection and correction of malapropisms - Hirst G et. al.

[19] Extended gloss overlaps as a measure of semantic relatedness - Banerjee S et. al.

[20] Using measures of semantic relatedness for word sense disambiguation - Patwardhan S et. al.

[21] On Kuhns Hungarian Method A tribute from Hungary - Andras Frank

[22] Inline::Perl - <http://search.cpan.org/pat/Inline-Java-0.52/Java.pod>

[23] ROUGE: a Package for Automatic Evaluation of Summaries - Chin-Yew Lin et. al.