Unsupervised Approach for Selecting Sentences in Query-based Summarization

Yllias Chali and Shafiq R. Joty
Department of Mathematics and Computer Science
University of Lethbridge
4401 University Drive
Lethbridge, Alberta, Canada, T1K 3M4
chali,jotys@cs.uleth.ca

Abstract
When a user is served with a ranked list of relevant documents by the standard document search engines, his search task is usually not over. He has to go through the entire document contents to judge its relevance and to find the precise piece of information he was looking for. Query-relevant summarization tries to remove the onus on the end-user by providing more condensed and direct access to relevant information. Query-relevant summarization is the task to synthesize a fluent, well-organized summary of the document collection that answers the user questions. We extracted several features of different types (i.e. lexical, lexical semantic, statistical and cosine similarity) for each of the sentences in the document collection in order to measure its relevancy to the user query. We experimented with two well-known unsupervised statistical machine learning techniques: K-Means and EM algorithms and evaluated their performances. For all these methods of generating summaries, we have shown the effects of different kinds of features.

Feature Extraction
The sentences in the document collection and topic narration are analyzed in various levels and each of the document-sentences is represented as a vector of feature-values. We use several types of features and investigate below their contribution to generate quality summary. The features can be divided into several categories:

Lexical Features
N-gram Overlap With the view to measure the N-gram (N=1,2,3,4) overlap scores, a Query Pool and a Sentence Pool are created. In order to create the Query Pool, we took the query sentences, for each query sentence we created a set of related sentences by replacing an important word (i.e. noun, verb, adverb and adjective) by its synonym(s). We created a sentence pool for each of the document-sentences in the same way. We measure the recall based n-gram scores using the following formula:
\[ N - \text{gram}(S, Q) = \frac{\sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)} \]
\[ n - \text{gramscore} = \arg\max_{i}(\arg\max_{j} N - \text{gram}(s_i, q_j)) \]

Where, \( n \) stands for the length of the \( n - \text{gram} \) \((n = 1, 2, 3, 4)\) and \( \text{Count}_{\text{match}}(\text{gram}_n) \) is the maximum number of \( n \)-grams co-occurring in the query and candidate sentence. \( q_j \) is the \( j^{th} \) sentence in the query pool and \( s_i \) is the \( i^{th} \) sentence in the sentence pool.

Longest Common Subsequence and Weighted Longest Common Subsequence

A sequence \( W = [w_1, w_2, ..., w_n] \) is a subsequence of another sequence \( X = [x_1, x_2, ..., x_m] \), if there exists a strict increasing sequence \([i_1, i_2, ..., i_k]\) of indices of \( X \) such that for all \( j = 1, 2, ..., k \) we have \( x_{i_j} = w_j \) (Cormen, Leiserson, & Rivest 1989). Given two sequences, \( S_1 \) and \( S_2 \), the longest common subsequence (LCS) of \( S_1 \) and \( S_2 \) is a common subsequence with maximum length.

The longer the LCS of two sentences is, the more similar the two sentences are. Following (Lin 2004), we used LCS-based F-measure to estimate the similarity between the document sentence \( S \) of length \( m \) and the query sentence \( Q \) of length \( n \) as follows:

\[ R_{lcs}(S, Q) = \frac{\text{LCS}(S, Q)}{m} \]
\[ P_{lcs}(S, Q) = \frac{\text{LCS}(S, Q)}{n} \]
\[ F_{lcs}(S, Q) = (1 - \alpha) \times P_{lcs} + \alpha \times R_{lcs} \]

Where, LCS(S,Q) is the length of a longest common subsequence of \( S \) and \( Q \) and \( \alpha \) is a constant that determines the importance of precision and recall. In order to measure LCS score for a sentence we took a similar approach as the previous section using both the question sentence and a query sentence. Following (Lin 2004), the LCS score between the document sentence \( S \) of length \( m \) and the query sentence \( Q \) of length \( n \) can be computed as follows:

\[ R_{skip_2}(S, Q) = \frac{\text{SKIP}_2(S, Q)}{C(m, 2)} \]
\[ P_{skip_2}(S, Q) = \frac{\text{SKIP}_2(S, Q)}{C(n, 2)} \]
\[ F_{skip_2}(S, Q) = (1 - \alpha) \times P_{skip_2} + \alpha \times R_{skip_2} \]

Where, \( \text{SKIP}_2(S, Q) \) is the number of skip bi-gram matches between \( S \) and \( Q \) and \( \alpha \) is a constant that determines the importance of precision and recall. \( C \) is the combination function. We call the Equation 3, skip bigram-based F-measure. We computed the skip bigram-based F-measure as follows:

\[ \text{Skip–bi–gramscore} = \arg\max_{i}(\arg\max_{j} F_{\text{skip}_2}(s_i, q_j)) \]

Head and Head Related-words Overlap

The number of heads common in between two sentences can indicate how much they are relevant to each other. In order to extract the heads from both query sentence and the sentence the sentence are parsed by Minipar\(^2\). From the parse trees we extract the heads and measure the overlap between them as follows:

\[ \text{Exact Head Score} = \frac{\sum_{w_1 \in \text{HeadSet}} \text{Count}_{\text{match}}(w_1)}{\sum_{w_1 \in \text{HeadSet}} \text{Count}(w_1)} \]

Where HeadSet is the set of head words in the sentence and \( \text{Count}_{\text{match}} \) is the number of matches between the HeadSet of query and sentence.

Again, we take the synonyms, hyponyms and hypernyms of both the query-head words and sentence-head words and measure the overlap similarly using the formula:

\[ \text{Head Related Score} = \frac{\sum_{w_1 \in \text{HeadRelSet}} \text{Count}_{\text{match}}(w_1)}{\sum_{w_1 \in \text{HeadRelSet}} \text{Count}(w_1)} \]

Where, HeadRelSet is the set of synonyms, hyponyms and hypernyms of head words in the sentence and \( \text{Count}_{\text{match}} \) is the number of matches between the head related set of query and sentence.

Basic Element Overlap Measure

We extracted Basic Elements (BEs) for the sentences in the document collection by using BE package distributed by ISI\(^3\). We used the standard BE-F breaker included in the BE package.

Once we get the BEs for a sentence, we computed the likelihood ratio (LR) for each BE following (Zhou, Lin, & Hovy 2005). The LR score of each BE is an information theoretic measure that represents the relative importance in the BE list from the document set that contains all the texts.

---

\(^2\)Available at http://www.cs.ualberta.ca/~lindek/minipar.htm

\(^3\)BE website: http://www.isi.edu/~cy/BE
LEXICAL SEMANTIC FEATURES

We form a set of words which we call query related words by taking the important words (i.e. nouns, adjectives, verbs and adverbs) from the query, their synonyms, hypernyms/hyponyms and important words from their gloss definitions. Synonym overlap measure is the overlap between the list of synonyms of the important words extracted from the candidate sentence and query related words. Hypernym/hyponym overlap measure is the overlap between the list of hypernyms and hyponyms of the nouns extracted from the sentence in consideration and query related words and gloss overlap measure is the overlap between the list of important words that are extracted from gloss definition of nouns in the sentence in consideration and query related words.

STATISTICAL SIMILARITY MEASURES

Statistical similarity measures are based on the co-occurrence of similar words in a corpus. We have used two statistical similarity measures:

Dependency-based similarity measure and Proximity-based similarity measure

Dependency-based similarity measure uses the dependency relations among words in order to measure the similarity (Lin 1998b). It extracts the dependency triples then uses statistical approach to measure the similarity. Proximity-based similarity measure is computed based on the linear proximity relationship between words only (Lin 1998a). It uses the information theoretic definition of similarity to measure the similarity.

We used the thesaurus provided by Dekang Lin4. Using the data, one can retrieve most similar words for a given word. The similar words are grouped into clusters. Note that, for a word there can be more than one cluster. Each cluster represents the sense of the word and its similar words for that sense.

For each query-word, we extract all of its clusters from the thesaurus. Now, in order to determine the right cluster for a query word, we measure the overlap score between the query related words (i.e. exact words, synonyms, hypernyms/hyponyms and gloss) and the clusters. The hypothesis is that, the cluster that has more words common with the query related words is the right cluster. We chose the cluster for a word which has the highest overlap score.

Once we get the clusters for the query words, we measured the overlap between the cluster words and the sentence words:

\[ M = \frac{\sum_{w_1 \in \text{SenWords}} \text{Count}_{\text{match}}(w_1)}{\sum_{w_1 \in \text{SenWords}} \text{Count}(w_1)} \]

Where, SenWords is the set of words for the sentence and \( \text{Count}_{\text{match}} \) is the number of matches between the Sentence Words and the cluster of similar words.

GRAPH-BASED SIMILARITY MEASURE

In (Erkan & Radev 2004), the concept of graph-based centrality is used to rank a set of sentences, in producing generic multi-document summaries. A similarity graph is produced for the sentences in the document collection. In the graph, each node represents a sentence. The edges between nodes measure the cosine similarity between the respective pair of sentences. The degree of a given node is an indication of how much important the sentence is. Once the similarity graph is constructed, the sentences are then ranked according to their eigenvector centrality. The LexRank performed well in the context of generic summarization. To apply LexRank to query-focused context, a topic-sensitive version of LexRank is proposed in (Otterbacher, Erkan, & Radev 2005). We followed a similar approach in order to calculate this feature. The score of a sentence is determined by a mixture model of the relevance of the sentence to the query and the similarity of the sentence to other high-scoring sentences. We capture this idea by the following mixture model:

\[ p(s|q) = d \times \frac{\text{rel}(s|q)}{\sum_{z \in C} \text{rel}(z|q)} + (1 - d) \times \frac{\sum_{v \in C} \text{sim}(s, v)}{\sum_{z \in C} \text{sim}(z, v)} \times p(v|q) \]

Where C is the set of all sentences in the collection and \( \text{rel}(s|q) \) is the relevance of sentence s to the query q and computed as follows:

\[ \text{rel}(s|q) = \sum_{w \in q} \log (t_{f_w,s} + 1) \times \log (t_{f_w,q} + 1) \times \text{idf}_w \]

\[ \text{idf}_w = \log \left( \frac{N + 1}{0.5 + s_{f_w}} \right) \]

Where \( t_{f_w,s} \) and \( t_{f_w,q} \) are the number of times w appears in s and q, respectively, \( \text{idf}_w \) is the IDF value of word w, N is the total number of sentences in the cluster and \( s_{f_w} \) is the number of sentences that the word w appears in. The value of the parameter d in equation 4, which we call “bias”, is a trade-off between two terms in the equation and is set empirically. For higher values of d, we prefer the relevance to the question to similarity to other sentences. The denominators in both terms are for normalization. We measure the cosine similarity weighted by word IDF’s as the similarity between two sentences in a cluster:

\[ \text{sim}(x, y) = \frac{\sum_{w \in x, y} t_{f_{w,x}} t_{f_{w,y}} \times (\text{idf}_w)^2}{\sqrt{\sum_{x, \in x} (t_{f_{w,x}} \times \text{idf}_w)^2} \sqrt{\sum_{y, \in y} (t_{f_{w,y}} \times \text{idf}_y)^2}} \]

4http://www.cs.ualberta.ca/ lindek/downloads.htm
Ranking Sentences and Summary Generation

The method used in order to select sentences for query-based summary extraction, is of type unsupervised learning, that means, before the learning begins, it is not known how many subsets (clusters) there are or how they are distinguished from each other. To start with, we experimented with two unsupervised learning techniques with the features extracted in the previous section: (a) K-means learning and (b) Expectation Maximization (EM) learning.

K-means Learning

We start with a set of initial cluster centers and go through several iterations of assigning each object to the cluster whose center is closest. After all objects have been assigned, we recompute the center of each cluster as the centroid or mean ($\mu$) of its members. The distance function we use is squared Euclidean distance instead of true Euclidean distance. Since the square root is a monotonically growing function squared Euclidean distance has the same result as the true Euclidean distance but the computation overload is smaller when the square root is dropped.

Once we have learned the means of the clusters using the K-means algorithm, our next task is to rank the sentences in the previous section: (a) K-means learning and (b) Expectation Maximization (EM) learning. Bayes’ law says:

$$P(\bar{x}) = \sum_{i=1}^{k} P(C = i)P(\bar{x}|C = i)$$

$$= \sum_{i=1}^{k} P(C = i)P(\bar{x}|\mu_i, \Sigma_i)$$

Where the total likelihood of model $\Theta$ with $k$ components given the observed data points, $X = x_1, \ldots, x_n$ is:

$$L(\Theta|X) = \prod_{i=1}^{n} P(C = j)P(x_i|\Theta_j)$$

$$= \prod_{i=1}^{n} \sum_{j=1}^{k} w_jP(x_i|\mu_j, \Sigma_j)$$

$$\Leftrightarrow \sum_{i=1}^{n} \log \sum_{j=1}^{k} w_jP(x_i|\mu_j, \Sigma_j)$$

where $P$ is the probability density function (i.e. eq 5). $\mu_j$ and $\Sigma_j$ are the mean and covariance matrix of component $j$ respectively. Each component contributes a proportion, $w_j$, of the total population, such that: $\sum_{j=1}^{k} w_j = 1$. Log likelihood can be used instead of likelihood as it turns the product into sum.

However, a significant problem with the EM algorithm is that it converges to a local maximum of the likelihood function and hence the quality of the result depends on the initialization. We experimented with one summary (for document number D0703A from DUC2007) by initializing the parameters as follows:

$$\mu_j = \text{rand}(1, \ldots, d) \ast \frac{\Sigma(DATA) \ast 10 + \mu(DATA)}{\Sigma_j}$$

$$w_j = 1/K$$

The highly variable nature of the results of the tests is reflected in the very inconsistent values for the total log likelihood (see figure 1) and the results of repeated experiments indicated that using random starting values for initial estimates of the means frequently gave poor results. There are two possible solutions to this problem.

In order to get good results from using random starting values, we will run the EM algorithm several times and choose the initial configuration for which we get the maximum log likelihood among all configurations. Choosing
the best one among several runs is very computer intensive process. So, to improve the outcome of the EM algorithm on gaussian mixture models it is necessary to find a better method of estimating initial means for the components. To achieve this aim, we explored the widely used “K-means” algorithm as a cluster (means) finding method. That means, the means found by K-means clustering above will be utilized as the initial means for EM and we calculate the initial covariance matrices using the unbiased covariance estimation procedure (eq:6).

Once the sentences are clustered by EM algorithm, we filter out the sentences which are not question-relevant by checking their probabilities, $P(q_{r}, | x_{i}, \Theta)$ where, $q_{r}$ denotes the cluster “question-relevant’. If for a sentence $x_{i}$, $P(q_{r}, | x_{i}, \Theta) > 0.5$ then $x_{i}$ is considered to be question-relevant.

Our next task is to rank the question-relevant sentences in order to include them in the summary. This can be done easily by defining a weight vector $w$ for the features and multiplying it with the feature vector $x_{i}$ representing the sentence. So, the rank of the sentence (or feature vector) $x_{i}$ is given by:

$$score_{i} = x_{i}^{T}w$$

**Redundancy Checking and Generating Summary**

Once the sentences are scored, the sentences deemed to be important are compared to each other and only those that are not too similar to other candidates are included in the final answer or summary. (Goldstein et al. 1999) observed this what they called “maximum-marginal-relevancy (MMR)”. Following (Hovy et al. 2006), we modeled this by BE overlap between an intermediate summary and a to-be-added candidate summary sentence. We call this overlap ratio $R$, where $R$ is between 0 and 1 inclusively. Setting $R = 0.7$ means that a candidate summary sentence $s$, can be added to an intermediate summary, $S$, if the sentence has a BE overlap ratio less than or equal to 0.7.

**Experimental Evaluation**

This section describes the results of experiments conducted using ROUGE automatic evaluation package (Lin 2004) on DUC 2007 dataset provided by NIST 5. We generated summaries for the 45 topics in DUC 2007 and experimented with the performance of EM and K-means and the contributions of different types of features in generating good summaries for each of the algorithms. To assess the contribution of different features, we grouped them into four classes. They are as follows:

**Lexical:** N-gram (N=1,2,3,4), LCS, WLCS score, skip bigram, head, head synonym and BE overlap.

**Lexical semantic:** Synonym, hypernym/hyponym, gloss, dependency-based similarity and proximity based similarity.

**Cosine similarity:** Graph-based similarity.

**All features:** All the features above.

**Discussion**

Table 1 to table 3 and table 4 to table 6 show the evaluation measures for K-means and EM learning respectively. From our experiments, it is obvious that our systems achieve better results when we include lexical semantic and cosine similarity features with the lexical features and as our lexical features include the ROUGE measures so summaries based on only lexical features achieve good results.

Table 7 shows the F-scores of all the ROUGE measures for one baseline system, the best system in DUC 2007 and our two learning algorithms taking all features. The baseline system generates summaries by returning all the leading sentences (up to 250 words) in the <TEXT> field of the most recent document(s).

The experiment results shows that EM outperforms the K-means algorithm and comparing with the DUC 2007 participants our systems achieve top scores.

**Conclusion and Future Work**

Our experiments show the following: (a) our approaches achieve promising results, (b) EM outperforms the K-means algorithm and (c) our systems achieve better results when we include lexical semantic and cosine similarity features with the lexical features. In this paper, we experimented with mainly fatures of type bag of words, later we would like to add the tree kernel based syntactic and shallow-semantic features and experiment their contribution for generating quality summary. As for this research problem, our training data were not labeled so we could not use supervised learning algorithms. Our future plan is to experiment with the supervised learning techniques (i.e. SVM, MAXENT) and see how do they perform for this problem. We have also the plan to decompose the complex questions into several simple questions before we find the similarity measures between the document sentence and the query sentence.

---

5National Institute of Standards and Technology
Table 1: ROUGE-2 measures for different feature combinations of K-means Learning

<table>
<thead>
<tr>
<th>Scores</th>
<th>Lexical</th>
<th>Lexical Semantic</th>
<th>Cosine Similarity</th>
<th>All Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.074488</td>
<td>0.076827</td>
<td>0.085583</td>
<td>0.074625</td>
</tr>
<tr>
<td>Precision</td>
<td>0.080886</td>
<td>0.084017</td>
<td>0.092514</td>
<td>0.080753</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.077543</td>
<td>0.080241</td>
<td>0.08891</td>
<td>0.077568</td>
</tr>
</tbody>
</table>

Table 2: ROUGE-L measures for different feature combinations of K-means Learning

<table>
<thead>
<tr>
<th>Scores</th>
<th>Lexical</th>
<th>Lexical Semantic</th>
<th>Cosine Similarity</th>
<th>All Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.337800</td>
<td>0.331936</td>
<td>0.346491</td>
<td>0.348463</td>
</tr>
<tr>
<td>Precision</td>
<td>0.366650</td>
<td>0.362165</td>
<td>0.373882</td>
<td>0.393244</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.351575</td>
<td>0.346288</td>
<td>0.359562</td>
<td>0.369502</td>
</tr>
</tbody>
</table>

Table 3: ROUGE-SU measures for different feature combinations of K-means Learning

<table>
<thead>
<tr>
<th>Scores</th>
<th>Lexical</th>
<th>Lexical Semantic</th>
<th>Cosine Similarity</th>
<th>All Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.131197</td>
<td>0.127442</td>
<td>0.139002</td>
<td>0.135857</td>
</tr>
<tr>
<td>Precision</td>
<td>0.154859</td>
<td>0.152065</td>
<td>0.162368</td>
<td>0.176021</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.141963</td>
<td>0.138527</td>
<td>0.149630</td>
<td>0.153353</td>
</tr>
</tbody>
</table>

Table 4: ROUGE-2 measures for different feature combinations of EM Learning

<table>
<thead>
<tr>
<th>Scores</th>
<th>Lexical</th>
<th>Lexical Semantic</th>
<th>Cosine Similarity</th>
<th>All Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.088564</td>
<td>0.079678</td>
<td>0.087056</td>
<td>0.084658</td>
</tr>
<tr>
<td>Precision</td>
<td>0.095792</td>
<td>0.087032</td>
<td>0.094192</td>
<td>0.091732</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.092018</td>
<td>0.083171</td>
<td>0.090462</td>
<td>0.088053</td>
</tr>
</tbody>
</table>

Table 5: ROUGE-L measures for different feature combinations of EM Learning

<table>
<thead>
<tr>
<th>Scores</th>
<th>Lexical</th>
<th>Lexical Semantic</th>
<th>Cosine Similarity</th>
<th>All Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.354123</td>
<td>0.326875</td>
<td>0.344994</td>
<td>0.348606</td>
</tr>
<tr>
<td>Precision</td>
<td>0.383792</td>
<td>0.359347</td>
<td>0.373265</td>
<td>0.377634</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.368281</td>
<td>0.343217</td>
<td>0.358481</td>
<td>0.362540</td>
</tr>
</tbody>
</table>

Table 6: ROUGE-SU measures for different feature combinations of EM Learning

<table>
<thead>
<tr>
<th>Scores</th>
<th>Lexical</th>
<th>Lexical Semantic</th>
<th>Cosine Similarity</th>
<th>All Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.145570</td>
<td>0.128454</td>
<td>0.138369</td>
<td>0.143064</td>
</tr>
<tr>
<td>Precision</td>
<td>0.171446</td>
<td>0.153288</td>
<td>0.162077</td>
<td>0.167845</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.157339</td>
<td>0.139623</td>
<td>0.149153</td>
<td>0.154467</td>
</tr>
</tbody>
</table>

Table 7: ROUGE F-measures for the learning algorithms, baseline system and DUC-2007 best system

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
<th>ROUGE-W</th>
<th>ROUGE-SU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.064900</td>
<td>0.310740</td>
<td>0.113810</td>
<td>0.112780</td>
</tr>
<tr>
<td>Best System</td>
<td>0.122850</td>
<td>0.405610</td>
<td>0.153600</td>
<td>0.174700</td>
</tr>
<tr>
<td>K-means</td>
<td>0.100875</td>
<td>0.368522</td>
<td>0.138643</td>
<td>0.152057</td>
</tr>
<tr>
<td>EM</td>
<td>0.100875</td>
<td>0.373738</td>
<td>0.139323</td>
<td>0.163230</td>
</tr>
</tbody>
</table>

References


