Multi-document summarization using A* search and discriminative training

Ahmet Aker   Trevor Cohn   Robert Gaizauskas

Department of Computer Science
University of Sheffield, Sheffield, S1 4DP, UK
{a.aker, t.cohn, r.gaizauskas}@dcs.shef.ac.uk

Abstract

In this paper we address two key challenges for extractive multi-document summarization: the search problem of finding the best scoring summary and the training problem of learning the best model parameters. We propose an A* search algorithm to find the best extractive summary up to a given length, which is both optimal and efficient to run. Further, we propose a discriminative training algorithm which directly maximises the quality of the best summary, rather than assuming a sentence-level decomposition as in earlier work. Our approach leads to significantly better results than earlier techniques across a number of evaluation metrics.

1 Introduction

Multi-document summarization aims to present multiple documents in form of a short summary. This short summary can be used as a replacement for the original documents to reduce, for instance, the time a reader would spend if she were to read the original documents. Following dominant trends in summarization research (Mani, 2001), we focus solely on extractive summarization which simplifies the summarization task to the problem of identifying a subset of units from the document collection (here sentences) which are concatenated to form the summary.

Most multi-document summarization systems define a model which assigns a score to a candidate summary based on the features of the sentences included in the summary. The research challenges are then twofold: 1) the *search* problem of finding the best scoring summary for a given document set, and

2) the *training* problem of learning the model parameters to best describe a training set consisting of pairs of document sets with model or reference summaries – typically human authored extractive or abstractive summaries.

Search is typically performed by a greedy algorithm which selects each sentence in decreasing order of model score until the desired summary length is reached (see, e.g., Saggion (2005)) or using heuristic strategies based on position in document or lexical clues (Edmundson, 1969; Brandow et al., 1995; Hearst, 1997; Ouyang et al., 2010).[1] We show in this paper that the search problem can be solved optimally and efficiently using A* search (Russell et al., 1995). Assuming the model only uses features local to each sentence in the summary, our algorithm finds the best scoring extractive summary up to a given length in words.

Framing summarization as search suggests that many of the popular training techniques are maximising the wrong objective. These approaches train a classifier, regression or ranking model to distinguish between good and bad sentences under an evaluation metric, e.g., ROUGE (Lin, 2004). The model is then used during search to find a summary composed of high scoring ('good') *sentences* (see for a review Ouyang et al. (2010)). However, there is a disconnect between the model used for training and the model used for prediction. In this paper we present a solution to this disconnect in the form of a training algorithm that optimises the full prediction model directly with the search algorithm intact. The training algorithm learns parameters such that

---

[1] Genetic algorithms have also been devised for solving the search problem (see, e.g., Riedhammer et al. (2008)), however these approaches do not guarantee optimality, nor are they efficient enough to be practicable for large datasets.

the best scoring *whole summary* under the model has a high score under the evaluation metric. We demonstrate that this leads to significantly better test performance than a competitive baseline, to the tune of 3% absolute increase for ROUGE-1, -2 and -SU4.

The paper is structured as follows. Section 2 presents the summarization model. Next in section 3 we present an A* search algorithm for finding the best scoring (argmax) summary under the model with a constraint on the maximum summary length. We show that this algorithm performs search efficiently, even for very large document sets composed of many sentences. The second contribution of the paper is a new training method which directly optimises the summarization system, and is presented in section 4. This uses the minimum error-rate training (MERT) technique from machine translation (Och, 2003) to optimise the summariser's output to an arbitrary evaluation metric. Section 5 describes our experimental setup and section 6 the results. Finally we conclude in section 7.

2 Summarization Model

Extractive multi-document summarization aims to find the most important sentences from a set of documents, which are then collated and presented to the user in form of a short summary. Following the predominant approach to data-driven summarisation, we define a linear model which scores summaries as the weighted sum of their features,

$$s(\mathbf{y}|\mathbf{x}) = \Phi(\mathbf{x}, \mathbf{y}) \cdot \lambda \quad , \quad (1)$$

where $\mathbf{x}$ is the document set, composed of k sentences, $\mathbf{y} \subseteq \{1 \ldots k\}$ are the set of selected sentence indices, $\Phi(\cdot, \cdot)$ is a feature function which returns a vector of features for the candidate summary and λ are the model parameters. We further assume that the features decompose with the sentences in the summary, $\Phi(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathbf{y}} \phi(x_i)$, and therefore the scoring function also decomposes along the same lines,

$$s(\mathbf{y}|\mathbf{x}) = \sum_{i \in \mathbf{y}} \phi(x_i) \cdot \lambda \quad . \quad (2)$$

While this assumption greatly simplifies inference, it does constrain the representative power of the model by disallowing global features, e.g., those which measure duplication in the summary.[2] Under this model, the search problem is to solve

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} s(\mathbf{y}|\mathbf{x}) \quad , \quad (3)$$

for which we develop a best-first algorithm using A* search, as described in section 3. The training challenge is to find the parameters, λ, to best model the training set. This is achieved by finding λ such that $\hat{\mathbf{y}}$ is similar to the gold standard summary according to an automatic evaluation metric, as described in section 4.

3 A* Search

The prediction problem is to find the best scoring extractive summary (see Equation 3) up to a given length, L. At first glance, this appears to be a simple problem that might be solved efficiently with a greedy algorithm, say by taking the sentences in order of decreasing score and stopping just before the summary exceeds the length threshold. However, the greedy algorithm cannot be guaranteed to find the best summary; to do so requires arbitrary backtracking to revise previous incorrect decisions.

The problem of constructing the summary can be considered a search problem in which we start with an empty summary and incrementally enlarge the summary by concatenating a sentence from our document set. The search graph starts with an empty summary (the starting state) and each outgoing edge adds a sentence to produce a subsequent state, and is assigned a score under the model. A goal state is any state with no more words than the given threshold. The summarisation problem is then equivalent to finding the best scoring path (summed over the edge scores) between the start state and a goal state.

The novel insight in our work is to use A* search (Russell et al., 1995) to solve the prediction problem. A* is a best-first search algorithm which can efficiently find the best scoring path or the n-best paths (unlike the greedy algorithm which is not optimal, and the backtracking variant which is not efficient). The search procedure requires a scoring function for each state, here $s(\mathbf{y}|\mathbf{x})$ from (2), and

---

[2]Our approach could be adapted to support global features, which would require changes to the heuristic for A* search to bound the score obtainable from the global features. This may incur an additional computational cost over a purely local feature model and perhaps also necessitate using beam search.

a heuristic function which estimates the additional score to get from a given state to a goal state. For the search to be optimal – guaranteed to find the best scoring path as the first solution – the heuristic must be *admissible*, meaning that it bounds from above the score for reaching a goal state. We present three different admissible heuristics later in this section, which bound the score with differing tightness and consequently different search cost.

Algorithm 1 presents A* search for our extractive summarisation model. Given a set of sentences to summary, a scoring and a heuristic function, it finds the best scoring summary. This is achieved by building the search graph incrementally, and storing each frontier state in a priority queue (line 1) which is sorted by the sum of the state's score and its heuristic. These states are popped off the queue (line 3) and expanded by adding a sentence, which is then added to the schedule (lines 8–14). We designate special finishing states using a boolean variable (the last entry in the tuple in lines 1, 7 and 12). Finishing states (with value *T*) denote ceasing to expand the summary, and consequently their scores do not include the heuristic component. Whenever one of these states is popped in line 2, we know that it outscores all competing hypotheses and therefore represents the optimal summary (because the heuristic is guaranteed to never underestimate the cost to a goal state from an unfinished state).[3] Note that in algorithm 1 we create the summary by building a list of sentence indices in sorted order to avoid spurious ambiguity which would unnecessarily expand the search space. The function length$(\mathbf{y}, \mathbf{x}) = \sum_{n \in \mathbf{y}} \text{length}(x_n)$ returns the length of sentences specified.

We now return to the problem of defining the heuristic function, $h(\mathbf{y}; \mathbf{x}, l)$ which provides an upper bound on the additional score achievable in reaching a goal state from state $\mathbf{y}$. We present three different variants of increasing fidelity, that is, that bound the cost to a goal state more tightly. Algorithm 2 is the simplest, which simply finds the maximum score per word from the set of unused sen-

---

Algorithm 1 A* search for extractive summarization.

Require: set of sentences, $\mathbf{x} = x_1, \ldots, x_k$
Require: scoring function $s(\cdot)$
Require: heuristic function $h(\cdot)$
Require: summary length limit L
1: schedule = $[(0, \emptyset, \text{F})]$ {priority queue of triples}
                                       {(A* score, sentence indices, done flag)}
2: **while** schedule $\neq$ [] **do**
3:     $v, \mathbf{y}, f \leftarrow \text{pop}(\text{schedule})$
4:     **if** f = T **then**
5:         **return y**   {success}
6:     **else**
7:         push(schedule, $(s(\mathbf{y}|\mathbf{x}), \mathbf{y}, \text{T})$)
8:         **for** $y \leftarrow [\max(\mathbf{y}) + 1, k]$ **do**
9:             $\mathbf{y}' \leftarrow \mathbf{y} \cup y$
10:            **if** length$(\mathbf{y}', \mathbf{x}) \leq L$ **then**
11:                $v' \leftarrow s(\mathbf{y}'|\mathbf{x}) + h(\mathbf{y}'; \mathbf{x}, l)$
12:                push(schedule, $(v', \mathbf{y}', \text{F})$)
13:            **end if**
14:        **end for**
15:    **end if**
16: **end while**

---

tences and then extrapolates this out over the remaining words available to the length threshold. In the algorithm, we use the shorthand $s_n = \phi(x_n) \cdot \lambda$ for sentence n's score, $l_n = \text{length}(x_n)$ for its length and $l_{\mathbf{y}} = \sum_{n \in \mathbf{y}} l_n$ for the total length of the current state (unfinished summary).

Algorithm 2 Uniform heuristic, $h_1(\mathbf{y}; \mathbf{x}, L)$

Require: $\mathbf{x}$ sorted in order of score/length
1: $n \leftarrow \max(\mathbf{y}) + 1$
2: **return** $(L - l_{\mathbf{y}}) \max\left(\frac{s_n}{l_n}, 0\right)$

The h_1 heuristic is overly simple in that it assumes we can 'reuse' a high scoring short sentence many times despite this being disallowed by the model. For this reason we develop an improved bound, h_2, in Algorithm 3. This incrementally adds each sentence in order of its score-per-word until the length limit is reached. If the limit is to be exceeded, the heuristic scales down the final sentence's score based on the fraction of words than can be used to reach the limit.

The fractional usage of the final sentence in h_2 could be considered overly optimistic, especially when the state has length just shy of the limit L. If the next best ranked sentence is a long one, then it will be used in the heuristic to over-estimate of the state. This is complicated to correct, and doing so exactly would require full backtracking which is intractable and would obviate the entire point of using A* search. Instead we use a subtle modification in h_3 (Alg. 4) which is equivalent to h_2 except in the

---

[3]To improve the efficiency of Algorithm 1 we make a small modification to avoid expanding every possible edge in step 8, of which there are $O(k)$ options. Instead we expand a small number (here, 3) at a time and defer the remaining items until later by inserting a special node into the schedule. These special nodes are represented using a third 'to-be-continued' state into the done flag.

Algorithm 3 Aggregated heuristic, $h_2(\mathbf{y}; \mathbf{x}, L)$

Require: $\mathbf{x}$ sorted in order of score/length
1: $v \leftarrow 0$
2: $l' \leftarrow l_{\mathbf{y}}$
3: **for** $n \in [\max(\mathbf{y}) + 1, k]$ **do**
4:    **if** $s_n \leq 0$ **then**
5:       **return** v
6:    **end if**
7:    **if** $l' + l_n \leq L$ **then**
8:       $l' \leftarrow l' + l_n$
9:       $v \leftarrow v + s_n$
10:    **else**
11:       **return** $v + \frac{l_n}{L - l'} s_n$
12:    **end if**
13: **end for**
14: **return** v

Figure 1: Example of the A* search graph created to find the two top scoring summaries of length ≤ 7 when summarising four sentences with scores of 4, 3, 2 and 1 respectively and lengths of 5, 4, 3 and 1 respectively. The h_1 heuristic was used and the score and heuristic scores are shown separately for clarity. Bold nodes were visited while dashed nodes were visited but found to exceed the length constraint.

instance where the next best score/word sentence is too long, where it skips over these sentences until it finds the best scoring sentence that does fit. This helps to address the overestimate of h_2 and should therefore lead to a smaller search graph and faster runtime due to its early elimination of dead-ends.

Algorithm 4 Agg.+final heuristic, $h_3(\mathbf{y}; \mathbf{x}, L)$

Require: $\mathbf{x}$ sorted in order of score/length
1: $n \leftarrow \max(\mathbf{y}) + 1$
2: **if** $n \leq k \wedge s_n > 0$ **then**
3:    **if** $l_{\mathbf{y}} + l_n \leq L$ **then**
4:       **return** $h_2(\mathbf{y}; \mathbf{x}, L)$
5:    **else**
6:       **for** $m \in [n + 1, k]$ **do**
7:          **if** $l_{\mathbf{y}} + l_m \leq L$ **then**
8:             **return** $s_m \frac{L - l_{\mathbf{y}}}{l_m}$
9:          **end if**
10:       **end for**
11:    **end if**
12: **end if**
13: **return** 0

The search process is illustrated in figure 1. When a node is visited in the search, if it satisfied the length constraint then the all its child nodes are added to the schedule. These nodes are scored with the score for the summary thus far plus a heuristic term. For example, the value of 4+1.5=5.5 for the $\{1\}$ node arises from a score of 4 plus a heuristic of $(7 - 5) \cdot \frac{3}{4} = 1.5$, reflecting the additional score that would arise if it were to use half of the next sentence to finish the summary. Note that in finding the best two summaries the search process did not need to instantiate the full search graph.

To test the efficacy of A* search with each of the different heuristic functions, we now present empirical runtime results. We used the training data as described in Section 5.2 and for each document set
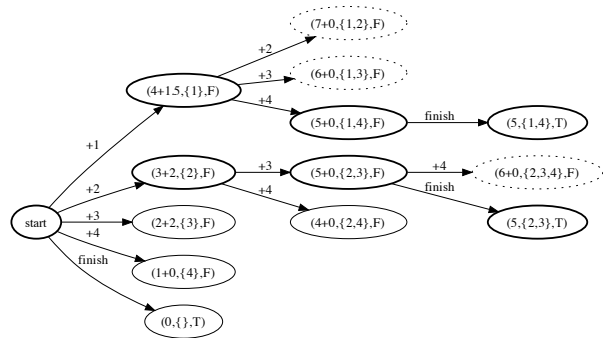
generated the 100-best summaries with word limit $L = 200$. Figure 2 shows the number of nodes and edges visited by A* search, reflecting the space and time cost of the algorithm, as a function of the number of sentences in the document set being summarised. All three heuristics shown an empirical increase in complexity that is roughly linear in the document size, although there are some notable outliers, particularly for the uniform heuristic. Surprisingly the aggregated heuristic, h_2, is not considerably more efficient than the uniform heuristic h_1, despite bounding the cost more precisely. However, the aggregated+final heuristic, h_3, consistently outperforms the other two methods. For this reason we have used h_3 in all subsequent experimentation.

4 Training

We frame the training problem as one of finding model parameters, λ, such that the predicted output, $\hat{\mathbf{y}}$ closely matches the gold standard, $\mathbf{r}$.[4] The quality of the match is measured using an automatic evaluation metric. We adopt the standard machine learning terminology of loss functions, which measure the degree of error in the prediction, $\Delta(\hat{\mathbf{y}}, \mathbf{r})$. In our case the accuracy is measured by the ROUGE

---

[4]The gold standard is typically an abstractive summary, and as such it is usually impossible for an extractive summarizer to match it exactly.
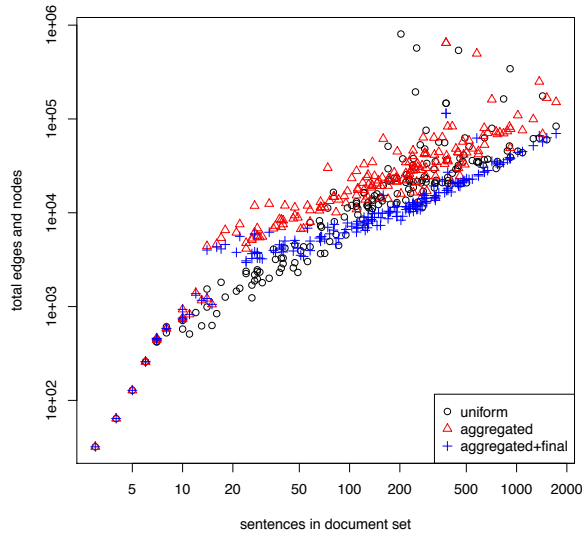
Figure 2: Efficiency of A* search search is roughly linear in the number of sentences in the document set. The y axis measures the search graph size in terms of the number of edges in the schedule and the number of nodes visited. Measured with the final parameters after training to optimise ROUGE-2 with the three different heuristics and expanding five nodes in each step.

score, R, and the loss is simply 1 - R. The training problem is to solve

$$\hat{\lambda} = \arg\min_{\lambda} \Delta(\hat{\mathbf{y}}, \mathbf{r}) \quad , \qquad (4)$$

where with a slight abuse of notation, $\hat{\mathbf{y}}$ and $\mathbf{r}$ are taken to range over the corpus of many document-sets and summaries.

To optimise the weights we use the minimum error rate training (MERT) technique (Och, 2003), as used for training statistical machine translation systems. This approach is a first order optimization method using Powell search to find the parameters which minimise the loss on the training data. MERT requires n-best lists which it uses to approximate the full space of possible outcomes. We use the A* search algorithm to construct these n-best lists,[5] and use MERT to optimise the ROUGE score on the training set for the R-1, R-2 and R-SU4 variants of the metric.

---

[5]We used $n = 100$ in our experiments.

5 Experimental settings

In this section we describe the features for which we learn weights. We also describe the input data used in training and testing.

5.1 Summarization system

The summarizer we use is an extractive, query-based multi-document summarization system. It is given two inputs: a query (place name) associated with an image and a set of documents. The summarizer uses the following features, as reported in previous work (Edmundson, 1969; Brandow et al., 1995; Radev et al., 2001; Conroy et al., 2005; Aker and Gaizauskas, 2009; Aker and Gaizauskas, 2010a):

- **querySimilarity**: Sentence similarity to the query (cosine similarity over the vector representation of the sentence and the query).
- **centroidSimilarity**: Sentence similarity to the centroid. The centroid is composed of the 100 most frequently occurring non stop words in the document collection (cosine similarity over the vector representation of the sentence and the centroid). For each word/term in the vector we store a value which is the product of the *term frequency* in the document and the *inverse document frequency*, a measurement of the term's distribution over the set of documents (Salton and Buckley, 1988).
- **sentencePosition**: Position of the sentence within its document. The first sentence in the document gets the score 1 and the last one gets $\frac{1}{n}$ where n is the number of sentences in the document.
- **inFirst5**: Binary feature indicating whether the sentence occurs is one of the first 5 sentences of the document.
- **isStarter**: A sentence gets a binary score if it starts with the query term (e.g. *Westminster Abbey*, *The Westminster Abbey*, *The Westminster* or *The Abbey*) or with the object type, e.g. *The church*. We also allow gaps (up to four words) between *the* and the query/object type to capture cases such as *The most magnificent abbey*, etc.
- **LMProb**: The probability of the sentence under a unigram language model. We trained a separate language model on Wikipedia articles about locations for each object type, e.g.,

church, *bridge*, etc. When we generate a summary about a location of type church, for instance, then we apply the church language model on the related input documents related to the location.[6]

- *sentenceCount*: Each sentence gets assigned a value of *1*. This feature is used to learn whether summaries with many sentences are better than summaries with few sentences or vice versa.
- *wordCount*: Number of words in the summary, to decide whether the model should favor long summaries or short ones.

5.2 Data

For training and testing we use the freely available image description corpus described in Aker and Gaizauskas (2010b). The corpus is based around 289 images of static located objects (e.g *Eiffel Tower*, *Mont Blanc*) each with a manually assigned place name and object type category (e.g. *church*, *mountain*). For each place name there are up to four model summaries that were created manually after reading existing image descriptions taken from the *VirtualTourist* travel community web-site. Each summary contains a minimum of 190 and a maximum of 210 words. We divide this set of 289 place names into training and testing sets. Both sets are described in the following subsections.

Training We use 184 place names from the 289 set for training feature weights. For each training place name we gather all descriptions associated with it from *VirtualTourist*. We compute for each sentence in each description a ROUGE score by comparing the sentence to those included in the model summaries for that particular place name and retaining the highest score. Table 1 gives some details about this training data.

We use ROUGE as a metric to maximize because it is also used in DUC[7] and TAC.[8] However, it should be noted that any automatic metric could be used instead of ROUGE. In particular we use ROUGE 1 (R-1), ROUGE 2 (R-2) and ROUGE SU4 (R-SU4). R-1 and R-2 compute the number

---

[6]For our training and testing sets we manually assigned each location to its corresponding object type (Aker and Gaizauskas, 2009).

[7]http://duc.nist.gov/

[8]http://www.nist.gov/tac/

	Max	Min	Avg
Sentences/place	1724	3	260
Words/sentence	37	3	17

Table 1: The training input data contains 184 place names with 42333 sentences in total. The numbers in the columns give detail about the number of sentences for each place and the lengths of the sentences.

	Max	Min	Avg
Documents/place	20	5	12
Sentences/place	1716	15	132
Sentences/document	275	1	10
Words/sentence	211	1	20

Table 2: In domain test data. The numbers in the columns give detail about the number of documents (descriptions) for each place, number of sentences for each place and document (description) and the lengths of the sentences.

of uni-gram and bi-gram overlaps, respectively, between the automatic and model summaries. R-SU4 allows bi-grams to be composed of non-contiguous words, with a maximum of four words between the bi-grams.

Testing For testing purposes we use the rest of the place names (105) from the 289 place name set. For each place name we use a set of input documents, generate a summary from these documents using our summarizer and compare the results against model summaries of that place name using ROUGE. We experimented with two different input document types: out of domain and in domain.

The in domain documents are the VirtualTourist original image descriptions from which the model summaries were derived. As with the training set we take all place name descriptions for a particular place and use them as input documents to our summarizer. Table 2 summarizes these input documents.

The out of domain documents are retrieved from the web. Compared to the in domain documents these documents should more challenging to summarize because they will contain different kinds of documents to those seen in training. For each place name we retrieved the top ten related web-documents using the Yahoo! search engine with the place name as a query. The text from these documents is extracted using an HTML parser and passed to the summarizer. Table 3 gives an overview of this data.

487

	Max	Min	Avg
Sentences/place	1773	55	328
Sentence/document	874	1	32
Words/sentence	236	1	21

Table 3: Out of domain test data. The numbers in the columns give detail about the number of sentences for each place and document and the lengths of the sentences.

Type	metric	R-1	R-2	R-SU4
wordLimit	R-1	**0.5792**	0.3176	0.3580
	R-2	0.5656	**0.3208**	0.3510
	R-SU4	0.5688	0.3197	**0.3585**
sentenceLimit	R-1	**0.5915**	0.3507	0.3881
	R-2	0.5783	**0.3601**	0.3890
	R-SU4	0.5870	0.3546	**0.3929**
regression	R-1	0.4993	0.1946	0.2448
	R-2	0.4833	0.1949	0.2413
	R-SU4	**0.5009**	**0.2031**	**0.2562**

Table 4: ROUGE scores obtained on the training data.

6 Results

To evaluate our approach we used two different assessment methods: ROUGE (Lin, 2004) and manual readability. In the following we present the results of each assessment.

6.1 Automatic Evaluation using ROUGE

We report results for training and testing. In both training and testing we distinguish between three different summaries: *wordLimit*, *sentenceLimit* and *regression*. *WordLimit* and *sentenceLimit* summaries are the ones generated using the model trained by MERT. As described in section 4 we trained the summariser using the A* search decoder to maximise the ROUGE score of the best scoring summaries. We used the heuristic function *h3* in A* search because it is the best performing heuristic, and 100-best lists. To experiment with different summary length conditions we differentiate between summaries with a word limit (*wordLimit*, set to 200 words) and summaries containing *N* number of sentences (*sentenceLimit*) as stop condition in A* search. We set *N* so that in both *wordLimit* and *sentenceLimit* summaries we obtain more or less the same number of words (because our training data contains on average 17 words for each word we set *N* to 12, 12*17=194). However, this is only the case in the training. In the testing for both *wordLimit* and *sentenceLimit* we generate summaries with the same word limit constraint which allows us to have a fair comparison between the ROUGE recall scores.

The *regression* summaries are our baseline. In these summaries the sentences are ranked based on the weighted features produced by Support Vector Regression (SVR).[9] Ouyang et al. (2010) use multi-document summarization and linear regression methods to rank sentences in the documents. As regression model they used SVR and showed

---
[9]We use the term *regression* to refer to SVR.

that it out-performed classification and Learning To Rank methods on the DUC 2005 to 2007 data. For comparison purpose we use SVR as a baseline system for learning feature weights. It should be noted that these weights are learned based on single sentences. However, to have a fair comparison between all our summary types we use these weights to generate summaries using the A* search with the word limit as constraint. We do this for reporting both for training and testing results.

The results for training are shown in Table 4. The table shows ROUGE recall numbers obtained by comparing model summaries against automatically generated summaries on the training data. Because in training we used three different metrics (R-1, R-2, R-SU4) to train weights we report results for each of these three different ROUGE metrics.

In Table 4 we can see that the scores for *wordLimit* and *sentenceLimit* type summaries are always at maximum on the metric they were trained on (this can be observed by following the main diagonal of the result matrix). This confirms that MERT is maximizing the metric for which it was trained. However, this is not the case for regression results. The scores obtained with R-SU4 metric trained weights achieve higher scores on R-1 and R-2 compared to the scores obtained using weights trained on those metrics. This is most likely due to SVR being trained on sentences rather than over entire summaries, and thereby not adequately optimising the metric used for evaluation.

The results for testing are shown in Tables 5 and 6. As with the training setting we report ROUGE recall scores. We use the testing data described in section 5.2 for this setting. However, because we have two different input document sets we report separate results for each of these (Table 5 shows result for in domain data and Table 6 shows result for out

Type	metric	R-1	R-2	R-SU4
wordLimit	R-1	**0.3733**	**0.0842**	0.1399
	R-2	0.3731	**0.0842**	**0.1402**
	R-SU4	0.3627	0.0794	0.1340
sentenceLimit	R-1	0.3664	0.0774	0.1321
	R-2	0.3559	0.0717	0.1251
	R-SU4	0.3629	0.0778	0.1312
regression	R-1	0.3431	0.0669	0.1229
	R-2	0.2934	0.0560	0.1043
	R-SU4	0.3417	0.0668	0.1226

Table 5: ROUGE scores obtained on the testing data. The automated summaries are generated using the in domain input documents.

Type	metric	R-1	R-2	R-SU4
wordLimit	R-1	**0.3758**	0.0882	0.1421
	R-2	0.3755	**0.0895**	**0.1423**
	R-SU4	0.369	0.0812	0.137
sentenceLimit	R-1	0.3541	0.0693	0.1226
	R-2	0.3426	0.0638	0.1157
	R-SU4	0.3573	0.073	0.1251
regression	R-1	0.3392	0.0611	0.1179
	R-2	0.3422	0.0606	0.1164
	R-SU4	0.3413	0.0606	0.1176

Table 6: ROUGE scores obtained on the testing data. The automated summaries are generated using the out of domain input documents.

of domain data). Again as with the training setting we report results for the different metrics (R-1, R-2, R-SU4) separately.

From Table 5 we can see that the *wordLimit* summaries score highest compared to the other two types of summaries. This is different from the training results where *sentenceLimit* summary type summaries are the top scoring ones. As mentioned earlier the *sentenceLimit* summaries contain exactly 12 sentences, where on average each sentence in the training data has 17 words. We picked 12 sentences to achieve roughly the same word limit constraint ($12 \times 17 = 204$) so they can be compared to the *wordLimit* and *regression* type summaries. However, these *sentenceLimit* summaries have an average of 221 words, which explains the higher ROUGE recall scores seen in training compared to testing (where a 200 word limit was imposed).

The *wordLimit* summaries are significantly better than the scores from the other summary types irrespective of the evaluation metric.[10] It should be

---
[10]Significance is reported at level $p < 0.001$. We used Wilcoxson signed ranked test to perform significance.

noted that these summaries are the only ones where the training and testing had the same condition in A* search concerning the summary word limit constraint. The scores in *sentenceLimit* type summaries are significantly lower than *wordLimit* summaries, despite using MERT to learn the weights. This shows that training the true model is critical for getting good accuracy. The *regression* type summaries achieved the worst ROUGE metric scores. The weights used to generate these summaries were trained on single sentences using SVR. These results indicate that if the goal is to generate high scoring summaries under a length limit in testing, then the same constraint should also be used in training.

From Table 5 and 6 we can see that the summaries obtained from VirtualTourist captions (in domain data) score roughly the same as the summaries generated using web-documents (out of domain data) as input. A possible explanation is that in many cases the VirtualTourist original captions contain text from Wikipedia articles, which are also returned as results from the web search. Therefore the web-document sets included similar content to the VirtualTourist captions.

6.2 Manual Evaluation

We also evaluated our summaries using a readability assessment as in DUC and TAC. DUC and TAC manually assess the quality of automatically generated summaries by asking human subjects to score each summary using five criteria – *grammaticality, redundancy, clarity, focus* and *coherence* criteria. Each criterion is scored on a five point scale with high scores indicating a better result (Dang, 2005).

For this evaluation we used the best scoring summaries from the *wordLimit* summary type (R-1, R-2 and R-SU4) generated using web-documents (out of domain documents) as input. We also evaluate the *regression* summary types generated using the same input documents to investigate the correlation between high and low ROUGE metric scores to manual evaluation ones. From the *regression* summary type we only use summaries under the *R2* and *RSU4* trained models.

In total we evaluated five different summary types (three from *wordLimit* and two from *regression*). For each type we randomly selected 30 place names and asked three people to assess the summaries for these place names. Each person was shown all 150

Criterion	wordLimit			regression	
	R1	R2	RSU4	R2	RSU4
clarity	4.03	3.92	3.99	3.00	2.92
coherence	3.31	3.06	2.99	2.12	1.88
focus	3.79	3.56	3.54	2.44	2.29
grammaticality	4.21	4.13	4.13	3.93	3.87
redundancy	4.19	4.33	4.41	4.47	4.44

Table 7: Manual evaluation results for the *wordLimit* (R1, R2, RSU4) and *regression* (R2, RSU4) summary types. The numbers in the columns are the average scores.

summaries (30 from each summary type) in a random way and was asked to assess them according to the DUC and TAC manual assessment scheme. The results are shown in Table 7.[11]

From Table 7 we can see that overall the *wordLimit* type summaries perform better than the *regression* ones. For each metric in *regression* summary types (R-2 and R-SU4) we compute the significance of the difference with the same metrics in *wordLimit* summary types.[12] The results for the *clarity*, *coherence* and *focus* criteria in *wordLimit* summaries are significantly better than in *regression* ones ($p < 0.001$) irrespective of the training metric. These results concur with the automatic evaluation results as described in section 6.1. However, this is not the case for the *grammaticality* and *redundancy* criteria. Although in *regression* type summaries the scores for the *grammaticality* criterion are lower than those in *wordLimit* summaries the difference is not significant. Furthermore, we can see that the *redundancy* scores for *regression* summaries are slightly higher than those for *wordLimit* summaries.

One reason for these differences might be the way we trained feature weights for *wordLimit* and *regression* summaries. As mentioned above, feature weights for *wordLimit* summaries are trained using summaries with a specific word limit constraint, whereas the weights for the *regression* summaries are learned using single sentences. Maximizing the ROUGE metrics using "final or output

---

[11]We computed the agreement between the users using intra class correlation with Cronbach's Alpha where the correlation coefficient ranges between 0 and 1. Numbers close to 1 indicate high correlation and numbers close to 0 indicate low correlation. For the clarity criterion the assessors' correlation coefficient is 0.547, for coherence 0.687, for focus 0.688, for grammaticality 0.232 and for redundancy 0.453.

[12]We compute significance test for the manual evaluation results using χ square.

like summaries" will lead to a higher content agreement between the training and the model summaries whereas this is not guaranteed with single sentences. With single sentences we have only a guarantee for high content overlap between single training and model sentences. However, when these sentences are combined into summaries it is not guaranteed that these summaries will also have high content overlap with the entire model ones. Therefore we believe if there is a high content agreement between the training and model summaries this could lead to more readable summaries. However, as we can see from Table 7 this hypothesis does not hold for all criteria. In case of the *redundancy* criterion we have compared to *wordLimit* summary type high scores in *regression* summaries although *wordLimit* summaries are significantly better than *regression* ones when it concerns the ROUGE scores. Thus it is likely that by aggressively optimising the ROUGE metric the model learns to game the metric, which does not penalise redundancy in the summaries. As such it may no longer possible to extrapolate trends from earlier correlation studies against human judgements (Lin, 2004).

To minimize redundancy in summaries it is necessary to also take into consideration global features addressing the linguistic aspects of the summaries. Furthermore, instead of ROUGE recall scores which do not take the repetition of information into consideration, ROUGE precision scores could be used as a metric in order to minimize the redundant content in the summaries.

7 Conclusion

In this paper we have proposed an A* search approach for generating a summary from a ranked list of sentences and learning feature weights for a feature based extractive multi-document summarization system. We developed an algorithm to learn optimize an arbitrary metric and showed that our approach significantly outperforms state of the art techniques. Furthermore, we highlighted the importance of uniformity in training and testing and argued that if the goal is to generate high scoring summaries under a length limit in testing, then the same constraint should also be used in training.

In this paper we experimented with sentence-local features. In the future we plan to expand this feature set with global features, especially ones mea-

suring lexical diversity in the summaries to reduce the redundancy in them. We will investigate various ways of incorporating these global features into our A* search. However this will incur an additional computational cost over a purely local feature model and therefore may necessitate using an approximate beam search. We also plan to investigate using other metrics in training in order to reduce redundant information in the summaries. Finally, we have made our summarizer publicly available as open-source software.[13]

References

A. Aker and R. Gaizauskas. 2009. Summary Generation for Toponym-Referenced Images using Object Type Language Models. *International Conference on Recent Advances in Natural Language Processing (RANLP) September 14-16, 2009, Borovets, Bulgaria*.

A. Aker and R. Gaizauskas. 2010a. Generating image descriptions using dependency relational patterns. *Proc. of the ACL 2010, Upsala, Sweden*.

A. Aker and R. Gaizauskas. 2010b. Model Summaries for Location-related Images. In *Proc. of the LREC-2010 Conference*.

R. Brandow, K. Mitze, and L.F. Rau. 1995. Automatic condensation of electronic publications by sentence selection* 1. *Information Processing & Management*, 31(5):675–685.

J.M. Conroy, J.D. Schlesinger, and J.G. Stewart. 2005. CLASSY query-based multi-document summarization. *Proc. of the 2005 Document Understanding Workshop, Boston*.

H.T. Dang. 2005. Overview of DUC 2005. *DUC 05 Workshop at HLT/EMNLP*.

H. Edmundson, P. 1969. New Methods in Automatic Extracting. *Journal of the Association for Computing Machinery*, 16:264–285.

M.A. Hearst. 1997. TextTiling: segmenting text into multi-paragraph subtopic passages. *Computational linguistics*, 23(1):33–64.

C-Y. Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out: Proc. of the ACL-04 Workshop*, pages 74–81.

I. Mani. 2001. *Automatic Summarization*. John Benjamins Publishing Company.

F.J. Och. 2003. Minimum error rate training in statistical machine translation. *Proc. of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, page 167.

Y. Ouyang, W. Li, S. Li, and Q. Lu. 2010. Applying regression models to query-focused multi-document summarization. *Information Processing & Management*.

D.R. Radev, S. Blair-Goldensohn, and Z. Zhang. 2001. Experiments in single and multi-document summarization using MEAD. *Document Understanding Conference*.

K. Riedhammer, D. Gillick, B. Favre, and D. Hakkani-T"ur. 2008. Packing the meeting summarization knapsack. *Proc. Interspeech, Brisbane, Australia*.

S.J. Russell, P. Norvig, J.F. Canny, J. Malik, and D.D. Edwards. 1995. *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, NJ.

H. Saggion. 2005. Topic-based Summarization at DUC 2005. *Document Understanding Conference (DUC05)*.

G. Salton and C. Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, 24(5):513–523.

---

[13]Available from `http://www.dcs.shef.ac.uk/~tcohn/a-star`