# Redundancy reduction for multi-document summaries using A* search and discriminative training

Ahmet Aker, Trevor Cohn and Robert Gaizauskas

University of Sheffield, UK

**Abstract.** In this paper we address the problem of optimizing global multi-document summary quality using A* search and discriminative training. Different search strategies have been investigated to find the globally best summary. In them the search is usually guided by an existing prediction model which can distinguish between good and bad summaries. However, this is problematic because the model is not trained to optimize the summary quality but some other peripheral objective. In this work we tackle the global optimization problem using A* search with the training of prediction model intact and demonstrate our method to reduce redundancy within a summary. We use the framework proposed by Aker et al. [1] as a baseline and adapt it to globally improve the summary quality. Our results show significant improvements over the baseline.

## 1 Introduction

Extractive multi-document summarization (MDS) aims to present the most important parts of multiple documents to the user in a condensed form [9, 13]. This is achieved by identifying a subset of sentences from the document collection which are concatenated to form the summary. Two common challenges in extractive MDS are: *search* – finding the best scoring summary from the documents – and *training* – learning the system parameters to best describe a training set consisting of pairs of documents and reference summaries.

In previous work the search problem is typically decoupled from the training problem. McDonald [14], for example, addresses the search problem by using Integer Linear Programming (ILP). In his ILP problem formulation he adopts the idea of Maximal Marginal Relevance (MMR) [5] to maximize the amount of relevant information in the summary and at the same time to reduce the redundancy within it. Others have also addressed the search problem using a variation of ILP [7, 8] but as well as using different approaches such as stack decoding algorithms [20], genetic algorithms [16] and submodular set function optimisation [12].

By separating search from training these approaches assume the existence of a predictive model which can distinguish between good and bad summaries. This is problematic because the model is not trained to optimize the summary quality but some other peripheral objective. The disconnect between the training and prediction settings compromises the predictive performance of the approach.

An exception is the work of Aker et al. [1], which proposes an integrated framework that trains the full prediction model directly with the search algorithm intact.

Their training algorithm learns parameters such that the best scoring *whole summary* under the model has a high score under an evaluation metric. However they only optimize the summary quality locally and do not take into account global features such as redundancy within the summary.

This paper addresses the redundancy problem within the integrated framework proposed by Aker et al. [1] and thus presents a novel approach to global optimization of summary quality. We present and evaluate our approach for incorporating a redundancy criterion into the framework. Our approach adapts the A* search to global optimization. The core idea of this approach is that redundant sentences are excluded from the summary if their redundancy with respect to the summary created so far exceeds a threshold. In our experiments this threshold is learned automatically from the data instead of being set manually as proposed in previous work.

The paper is structured as follows. Section 2 presents the work of Aker et al., [1], in detail. In Section 3 we describe our modifications to the framework proposed by Aker et al. and our proposed approach to address redundancy in extractive summarization. Section 4 describes our experimental setup to evaluate the proposed approach, and Section 5 the results. Finally, we conclude in Section 6.

## 2 Background

In this section we first review the work of Aker et al. [1] in detail, which is essential for the understanding of our modifications to their framework.

### 2.1 Summarization Model

A summarization model is used to score summaries. Summaries are ranked according to these scores, so that in search, the summary with the highest score can be selected. Aker et al. use the summarization model $s$ to score a summary:

$$s(\mathbf{y}|\mathbf{x}) = \sum_{i \in \mathbf{y}} \phi(x_i)\lambda \tag{1}$$

where $\mathbf{x}$ is the document set, composed of k sentences, $\mathbf{y} \subseteq \{1 \dots k\}$ is the set of indexes selected for the summary, $\phi(\cdot)$ is a feature function that returns a set of features values for each candidate summary and $\lambda$ is the weight vector associated with the set of features. In search we use the summarization model to find the maximum summary $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} s(\mathbf{y}|\mathbf{x}) \tag{2}$$

### 2.2 Search

In Aker et al. the creation of a multi-document summary is formulated as a search problem in which the aim is to find a subset of sentences from the entire set to form a summary. The search is also constrained so that the subset of sentences does not exceed the summary length threshold. In search, a search graph is constructed with edges representing the connections between the sentences and states with summaries.

Each node is associated with the information about the summary length and summary score. The authors start with an empty summary (start state) with length 0 and score 0 and follow an outgoing edge to expand it. A new state is created when a new sentence is added to the summary. The new state's length is updated with the number of words of the new sentence. The score of the state is computed under the summarization model described in the previous section. A goal state is any state or summary where it is not possible to add another sentence without exceeding the summary length threshold. The summarization problem is then finding the best scoring path (sum over the sentence scores on this path) between the start state and a goal state.

Aker et al. use the A* search algorithm [17] to efficiently traverse the search graph and accurately find the best scoring path. In A* search a best-first strategy is applied to traverse the graph from a starting state to a goal state. The search requires a scoring function for each state, here $s(\mathbf{y}|\mathbf{x})$ from Equation 1, and a heuristic function that estimates the additional score to get from a given state to a goal state. The search algorithm is guaranteed to converge to the optimal solution if the heuristic function is *admissible*, that is, if the function used to estimate the cost from the current node to the goal never overestimates the actual cost. The authors propose different heuristics with different run-time performances. The reported best performing heuristic is the "final aggregated heuristic". We use this heuristic as baseline and for our modification purposes.

### 2.3 Training

In Aker et al. training problem is formulated as one of finding model parameters, $\lambda$, such that the predicted output, $\hat{\mathbf{y}}$ closely matches the gold standard, $\mathbf{r}$. The quality of the match is measured using ROUGE [10]. In the training the standard machine learning terminology of loss functions, which measure the degree of error in the prediction, $\Delta(\hat{\mathbf{y}}, \mathbf{r})$ is adopted. The loss is formulated as $1 - R$ with $R$ as being the ROUGE score. The training problem is to solve

$$\lambda = \arg\min_{\lambda} \Delta(\hat{\mathbf{y}}, \mathbf{r}) \tag{3}$$

where $\hat{\mathbf{y}}$ and $\mathbf{r}$ are taken to range over the corpus of many document-sets and summaries. The prediction model is trained using the minimum error rate training (MERT) technique [15]. MERT is a first order optimization method using Powell search to find the parameters which minimize the loss on the training data [15]. MERT requires $n$-best lists which it uses to approximate the full space of possible outcomes. A* search is used to construct these $n$-best lists and MERT to optimize the objective metric such as ROUGE that is used to measure the summary quality.

## 3 Addressing redundancy

To address redundancy within a summary we adopt the framework of Aker et al. [1] described in the previous section in that we re-use their summarization and training of the prediction model.

### 3.1 A* search with redundancy reduction

In this section we present our approach to dealing with redundancy within multi-document summaries, which implement the idea of omitting or *jumping over* redundant sentences when selecting summary-worthy sentences from the input documents. When sentences from the input documents are merged and sorted in a list according to their summary-worthiness, the generation of a summary starts by first including a top summary-worthy sentence into the summary, then the next one until a desired summary length is reached. If a sentence from the list is found to be similar to the ones already included in the summary (i.e. to be redundant), then this sentence should not be included into the summary, but rather *jumped over*. We integrate the idea of *jumping over* redundant sentences into the A* search algorithm described by Aker et al. The difference between our implementation and the one of Aker et al. is the integration of a function jump$(\mathbf{y}, y)$ into the search process. We use this function to jump over a sentence with the index $y$ when it is redundant with respect to the summary $\mathbf{y}$. Thus compared to Aker et al. we do not only skip a sentence if it is too long as it is the case in Aker et al., but also when it is redundant compared to the summary created so far. In our work we replace the jump conditions of Aker et al. with:

$$\text{lengthConstraintsOK} \wedge \text{jump}(\mathbf{y}, y) == F \tag{4}$$

where lengthConstraintsOK represents the situation when the next sentence does not violate the summary length in Aker et al. and jump$(\mathbf{y}, y) == F$ the case where the next sentence is not redundant and therefore not to be jumped over.

*Jump based on redundancy threshold (JRT):* We use the similarity score of a sentence $x_i$ with respect to the summary $\mathbf{y}$ and a similarity or redundancy threshold $R$ to decide whether to jump over the sentence or not. In general we jump over a sentence $x_i$ if its similarity score is above $R$ (see Algorithm in 1). The similarity scores are computed using the sim$(.,.)$ function shown in Equation 5.

**Algorithm 1** Jump when similarity score is above a threshold $R$, $jump(\mathbf{y}, x_i)$

**Require:** require a similarity or redundancy threshold $R$
1: **if** sim$(\mathbf{y}, x_i) \leq R$ **then**
2: **return** $F$
3: **end if**
4: **return** $T$

$$sim(\mathbf{y}, x_j) = \frac{1}{n} \sum_{l=1}^{n} \frac{|ngrams(\mathbf{y}, l) \bigcap ngrams(x_j, l)|}{|ngrams(x_j, l)|} \tag{5}$$

where ngrams$(\mathbf{y}, n)$ is the set of n-grams in summary $\mathbf{y}$ and ngrams$(x_j, n)$ in sentence $x_j$ respectively. This method returns 0 if $\mathbf{y}$ and $x_j$ do not share any n-grams. When all n-grams of $x_j$ are found in the list of n-grams of $\mathbf{y}$ the method returns 1. Note that we use this function to only see how many n-grams of $x_j$ are found in $\mathbf{y}$. The other

direction is less important for our purpose. The idea of omitting redundant sentences if their redundancy score exceeds a threshold has already been introduced in previous work [4, 11, 18, 19]. However, in contrast to these studies, in which the redundancy threshold is set manually, we learn it automatically.

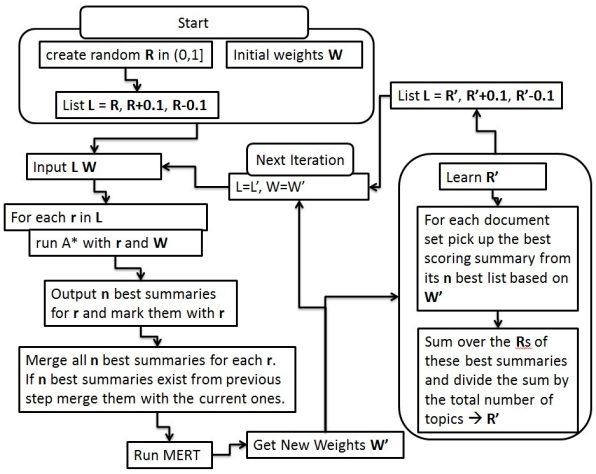


**Fig. 1.** Learning the redundancy threshold $R$. The learning procedure starts in the box denoted with *Start*.

To learn the redundancy threshold $R$ we make use of the entire framework (search and training) and proceed as shown in Figure 1. In the beginning (the top left of the figure) we create a random $R \in (0, 1]$. In addition to this $R$ we generate two further values: $R + 0.1 \leq 1$ and $R - 0.1 > 0$. These two additional numbers are used to move $R$ towards its optimum value. All three $R$s are used to generate $n$ best summaries using A* search. In the A* search we also require a prediction model to score the sentences. For this we start with an initial prediction model (initial feature weights $W$). For each of the $R$ values (denoted with $r$ in the figure) we then create an $n$ best list using A* search leading to $3 \times n$ summaries. If there are summaries from a previous step we extend the new $n$ best list with them, so that in training the entire history of $n$ best lists is provided. For each summary its corresponding $R$ value is known. Next, these $n$ best summaries are input to MERT to train new weights $W'$, i.e. a new prediction model. After obtaining $W'$ we can pick up the summary from the $n$ best summaries created for each document set MERT has used to come up with $W'$. We sum the $R$ values of those summaries (in total $m$ for $m$ document sets) and divide the sum by $m$ to obtain the new $R'$. We replace $R$ with $R'$ and $W$ with $W'$ and repeat the entire process until no new summaries are added to the $n$ best list, when the process stops. Depending on which $R$ was used to generate the best summaries ($R$, $R + 0.1$ or $R - 0.1$), the optimal value for $R$ (($R$ that leads to best summaries under the ROUGE metric)) will choose its direction either towards $> 0$ or $\leq 1$.

## 4 Experimental settings

In this section we describe the data used in the experiments, our summarization system and the training and testing procedure.

### 4.1 Data

For training and testing we use the freely available image corpus described in [3]. The corpus contains 296 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 extracted manually from existing image descriptions taken from the *VirtualTourist* travel community website. Each summary contains a minimum of 190 and a maximum of 210 words.

### 4.2 Summarization system

To generate summaries for each of the 296 document sets we use an extractive, query-based multi-document summarization system. It is given three inputs: a query (place name, e.g. *Westminster Abbey*), the object type associated with an image (e.g. *church*) and a set of web-documents retrieved using the place name as query. The summarizer uses the following features described in [2, 1]:

- ***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 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*) or with the object type, e.g. *The church*.
- ***LMProb***: The probability of the sentence under a bi-gram 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.[1]
- ***DepSim***: Similar to *LMProb* we trained a separate dependency pattern model using Wikipedia articles about locations for each object type. As in *LMProb* we use these models to score the input sentences. A sentence is scored based on the number of patterns it contains from the model.
- ***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.

[1] For our training and testing sets we manually assigned each location to its corresponding object type.

**Table 1.** ROUGE scores. In each row the results were obtained with the prediction model trained on the metric of that row.

| Recall | Aker et al. [1] | JRT |
|---|---|---|
| R2 | 0.094 | **0.109**∗ |
| RSU4 | 0.146 | **0.167**∗ |

**Table 2.** Example summary about the query *Akershus Castle*.

| |
|---|
| Norwegian Royalty have been buried in the Royal Mausoleum in the castle. During the 17th and 18th century the castle fell into decay, and restoration work only started in 1899. The Akershus castle and fortress are located on the eastern side of the Oslo harbor. The fortress was first used in battle in 1306. The original Akershus Castle is located inside the fortress. Akershus Fortress (Norwegian: Akershus Festning) is the old castle built to protect Oslo, the capital of Norway. The fortress was built in 1299, and the meaning of the name is 'the (fortified) house of (the district) Aker'. In the 1600s a castle (or in norsk, "slott") was built. In the reign of Christian IV the medieval stronghold was converted into a Renaissance castle and the fortifications were extended. Guided tours of the fortress in the summer, all year on request. The services are announced in the newspapers and are open to all. During World War II, several people were executed here by the German occupiers. The fortress was reconstructed several times to withstand increasing fighting power. The castle is well positioned overlooking Oslo's harbour. The fortress was strategically important for Oslo and therefore for Norway as well. |

## 5 Results

We use 191 document sets for training and 105 for testing. When training the prediction model we use ROUGE as a metric to maximize because it is also used for automatic summary evaluation in DUC[2] and TAC.[3] In particular, following DUC and TAC we use ROUGE 2 (R-2) and ROUGE SU4 (R-SU4) for both in training and testing. R-2 computes the number of bi-gram overlaps between the automatic and model summaries. R-SU4 measures uni-gram overlaps between two text units but also bi-grams composed of non-contiguous words, with a maximum of four words between the words. The results of our experiments are shown in Table 1.

As shown in Table 1 the results achieved with the $JRT$ method where we learn a redundancy threshold $R$ automatically are better than the ones obtained using the setting without the idea of jump. The $JRT$ method significantly[4] ($p < 0.001$) outperforms the method of Aker et al..[5]

The values of the learnt redundancy threshold $R$ differ for different ROUGE metrics: for R2 this is $0.5338$ and for RSU4 $0.4675$. The different $R$ values are expected given the different properties of R2 and RSU4. Compared to R2 the redundancy threshold for RSU4 is more strict which reflects the way RSU4 works. As mentioned in Section 4, RUS4 measure the uni-gram overlap between two text units but also bi-grams where gaps of up to four words are allowed between the words. This means that RSU4 is able to capture more similarities between sentences than R2, where single word overlaps are not captured. In R2 gaps within a bi-gram are allowed. For example bi-grams

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

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

[4] We use a two-tail paired T-test to compute significance test.

[5] We have also studied different alternative methods to the $JRT$ one to be used in the jump$(.,.)$ function such as favoring the following sentence to the current one if it is less redundant than the current one or combining the redundancy scores with the actual raw scores of the sentences and jumping only over the current sentence if the combined score is less than the combined score of the following sentence. However, the results by these alternative methods led only to moderate improvement over the baseline. For this reason we do not report those results.

**Table 3.** Readability evaluation results: Each cell shows the percentage of summaries scoring the ranking score heading the column for each criterion in the row as produced by the summary method indicated by the subcolumn heading – Aker et al. ($RW$) and $JRT$. The numbers indicate the percentage values averaged over the three people.

| | 5 | | 4 | | 3 | | 2 | | 1 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Criterion | RW | JRT | RW | JRT | RW | JRT | RW | JRT | RW | JRT |
| clarity | 6.2 | 22.4 | 41.7 | 73.5 | 29.2 | 2.0 | 20.8 | 0 | 2.1 | 2.0 |
| coherence | 6.2 | 28.6 | 18.8 | 42.9 | 33.3 | 24.5 | 37.5 | 4.1 | 4.2 | 0 |
| focus | 6.2 | 26.5 | 33.3 | 61.2 | 29.2 | 12.2 | 29.2 | 0 | 2.1 | 0 |
| grammar | 4.2 | 12.2 | 58.3 | 67.3 | 12.5 | 4.1 | 20.8 | 14.3 | 4.2 | 2.0 |
| redundancy | 4.2 | 8.2 | 8.3 | 61.2 | 2.1 | 12.2 | 41.7 | 18.4 | 43.8 | 0 |

**Table 4.** Readability evaluation results: Each cell shows the percentage of summaries scoring the ranking score $>= 4$ for each criterion in the row as produced by the summary method indicated by column heading – Aker et al. ($RW$) and $JRT$. The numbers indicate the percentage values averaged over the three people.

| Criterion | RW | JRT |
|---|---|---|
| clarity | 47.9 | 95.9 |
| coherence | 25 | 71.5 |
| focus | 39.5 | 87.7 |
| grammar | 30.2 | 79.5 |
| redundancy | 12.5 | 69.4 |

$AB$ and $A??B$ are identical in RSU4, but not in R2. Consequently, a stricter redundancy threshold is required in RSU4 than in R2. This fact illustrates also that there cannot be a single $R$ for every ROUGE metric and highlights the importance of learning it for each of the ROUGE metrics separately.

From the example summary about the query *Akershus Castle* shown in Table 2 we can see that the summary does capture a variety of facts about the castle such as when the castle was built, where it is located, etc. This type of essential information about the castle occurs only once in the summary. What is repeated in most of the sentences are referring expressions such as the name of the place (*Akershus Castle*) or the object type (*the castle* or *the fortress*). Sentences containing referring expressions are more likely to contain relevant information about the castle in the model summaries than sentences which do not contain such expressions. The redundancy thresholds are set to allow some repetition in the summary, which means that MERT learned to allow referring expressions to be repeated in the summary, so it can maximize the ROUGE metrics.

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 *structure*. Each criterion is scored on a five point scale with high scores indicating a better result [6]. In the evaluation we asked three people to assess the summaries. Each person was shown 100 summaries (50 from each summary type selected randomly from the entire test set of 105 places). The summaries were shown in a random way. The results of the manual evaluation are shown in Table 3. Table 4 shows percentage values of summaries which achieved scores at levels four or above.

We see from Table 3 that $JRT$ type summaries perform much better than in the Aker et al. setting where summaries are generated without redundancy detection. The percentage values at levels 5 and 4 (see Table 4) show that the $JRT$ summaries have more clarity (95.9% of the summaries), are more coherent (71.5% of the summaries), have better focus (87.7% of the summaries) and grammar (79.5% of the summaries) and contain less redundant information (69.4% of the summaries) than the ones generated in the $wordLimit$ setting (47.9%, 25%, 39.5%, 30.2% and 12.5%). The substantial improvement in redundancy from the Aker et al. setting to JRT demonstrated that incorporating a jump into a summarization system adds to redundancy reduction but also improves other quality aspects of the summary.

## 6 Conclusion

In this paper we proposed and evaluated an automatic method for improving the global quality of extractive multi-document summaries by means of reducing the redundancy within summaries. We used the framework proposed by Aker et al. [1] as a baseline because it uses a combined search and training approach to maximize the summary quality locally and adapted it for global optimization. We demonstrated that our proposed method, $JRT$, for redundancy reduction improves the quality of the summary over the baseline as indicated by the ROUGE metric and manual evaluation. In $JRT$ we jump over sentences which are more similar than a similarity threshold $R$ learnt automatically. We have seen that the properties of different ROUGE metrics require different redundancy thresholds, so that $R$ must be learned for each ROUGE metric separately. The automatically determined $R$ values appeared to be neither too strict nor too generous as they allow referring expressions to be redundant in the output summary but not whole factual assertions. This reflects the fact that in the model summaries the sentences containing referring expressions are also those which contain the most relevant information about a query.

In future work we intend to address several issues arising from this work. First, we intend to incorporate semantic knowledge into computation of the redundancy scores. Currently, when learning the $R$ value we purely use surface level comparison and compute the redundancy score between a sentence and a summary using uni and bi-gram lexical overlaps. By doing this we can only capture the repetition of information units if they are expressed in the same way. We believe that the results can be further improved if techniques to detect semantic overlaps are also used. Second, we aim to address the issue of information flow, which is currently missing in the output summaries. From the example summary we can see that the summary reads like the bag of sentences. By integrating flow into the A* search algorithm we hope to improve the readability of the summaries.

## References

1. Aker, A., Cohn, T., Gaizauskas, R.: Multi-document summarization using A* search and discriminative training. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. pp. 482–491. Association for Computational Linguistics (2010)

2. Aker, A., Gaizauskas, R.: Generating image descriptions using dependency relational patterns. Proc. of the ACL 2010, Upsala, Sweden (2010)
3. Aker, A., Gaizauskas, R.: Model Summaries for Location-related Images. In: Proc. of the LREC-2010 Conference (2010)
4. Barzilay, R., McKeown, K., Elhadad, M.: Information fusion in the context of multi-document summarization. In: Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics. pp. 550–557. Association for Computational Linguistics (1999)
5. Carbonell, J., Goldstein, J.: The use of mmr, diversity-based reranking for reordering documents and producing summaries. In: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. pp. 335–336. ACM (1998)
6. Dang, H.: Overview of DUC 2005. DUC 05 Workshop at HLT/EMNLP (2005)
7. Gillick, D., Favre, B.: A scalable global model for summarization. In: Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing. pp. 10–18. Association for Computational Linguistics (2009)
8. Gillick, D., Riedhammer, K., Favre, B., Hakkani-Tür, D.: A global optimization framework for meeting summarization. In: Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on. pp. 4769–4772. IEEE (2009)
9. Jones, K.: Automatic summarizing: factors and directions. Advances in Automatic Text Summarization pp. 1–12 (1999)
10. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out: Proc. of the ACL-04 Workshop pp. 74–81 (2004)
11. Lin, C., Hovy, E.: From single to multi-document summarization: A prototype system and its evaluation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. pp. 457–464. Association for Computational Linguistics (2002)
12. Lin, H., Bilmes, J.: Multi-document summarization via budgeted maximization of submodular functions. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 912–920. Association for Computational Linguistics (2010)
13. Mani, I., Maybury, M.: Advances in automatic text summarization. the MIT Press (1999)
14. McDonald, R.: A study of global inference algorithms in multi-document summarization. Advances in Information Retrieval pp. 557–564 (2007)
15. Och, F.: Minimum error rate training in statistical machine translation. Proc. of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1 p. 167 (2003)
16. Riedhammer, K., Gillick, D., Favre, B., Hakkani-Tür, D.: Packing the meeting summarization knapsack. Proc. Interspeech, Brisbane, Australia (2008)
17. Russell, S., Norvig, P., Canny, J., Malik, J., Edwards, D.: Artificial intelligence: a modern approach. Prentice hall Englewood Cliffs, NJ (1995)
18. Saggion, H.: A robust and adaptable summarization tool. Traitement Automatique des Langues 49(2) (2008)
19. Sauper, C., Barzilay, R.: Automatically generating wikipedia articles: A structure-aware approach. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1. pp. 208–216. Association for Computational Linguistics (2009)
20. Yih, W., Goodman, J., Vanderwende, L., Suzuki, H.: Multi-document summarization by maximizing informative content-words. In: Proceedings of IJCAI. vol. 7 (2007)