MultiSum
Query-Based Multi-Document Summarisation

Michael Rosner
Dept. Artificial Intelligence
University of Malta
mike.rosner@um.edu.mt

Carl Camilleri
Dept. Artificial Intelligence
University of Malta
ccam0002@um.edu.mt

Abstract

This paper describes a generic, open-domain multi-document summarisation system which combines new and existing techniques in a novel way. The system is capable of automatically identifying query-related online documents and compiling a report from the most useful sources, whilst presenting the result in such a way as to make it easy for the researcher to look up the information in its original context.

1 Introduction

Although electronic resources have several inherent advantages over traditional research media, they also introduce several drawbacks, such as *Information Overload* (Edmunds and Morris, 2000),which has become synonymous with the information retrieval phase of any research-related task. Another problem which is directly related to the one just described is that of *Source Identification* (Eppler and Mengis, 2004). This refers to the problem of having relevant results intermingled with results that are less relevant, or actually irrelevant.

Lastly, the researcher usually has to also manually traverse the relevant sources of information in order to form an answer to the research query.

These problems have led to the study of various areas in computing, all of which aim to try and minimise the manual effort of information retrieval and extraction, one of which is Multi-Document Summarisation (MDS).

The core aim of any MDS system is that of processing multiple sources of information and outputting a relatively brief but broad report or summary. Uses of MDS systems vary widely, from summarisation of closed-domain documents, such as news documents (Evans et al., 2004), to aggregation of information from several sources in an open domain.

2 Aims and Objectives

MDS techniques can be used in various tools that may help addressing the problems described in Section 1. On the other hand, a brief study of the relevant literature indicates that the majority of the work done in this area concerns closed-domains such as news summarisation, which is perhaps the reason why such tools have not yet become more popular. The objectives of this study are thus twofold.

- The primary objective is that of designing, implementing and evaluating an open-domain, query-based MDS system which is capable of compiling an acceptably-coherent report from the most relevant online sources of information, whilst making it easy for the reader to access the full source of information in its original context.

- A secondary objective of this study is Search Engine Optimisation (SEO): We require the system to produce summaries which, if published on the Internet, would be deemed relevant to the original query by search engine ranking algorithms. This is measured by keyoword density in the summary. Success on this objective addresses the problem of Source Identification since the summary would at the very least serve as a gateway to the other relevant sources from which it was formed.

Unsurprisingly, one of the problems that has to be overcome in the field of summarisation and particularly in an open-domain system such as ours is

© 2008. Licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported* license (http://creativecommons.org/licenses/by-nc-sa/3.0/). Some rights reserved.

Coling 2008: Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization, pages 25–32
Manchester, August 2008

the quality of output, as measured by a number of different linguistic and non-linguistic criteria (see Section 5). We have adopted a number of novel techniques to address this such as

- Multi-Layered Architecture
- Sentence Ordering Model
- Heuristic Sentence Filtering
- Paragraph Clustering

3 Background

3.1 Search Engine Ranking Criteria

Search engine ranking algorithms vary, and are continuously being optimised in order to provide better and more accurate results. However, some guidelines that outline factors which web masters need to take into account have been established (cf. Google (2007), Vaughn (2007)).

When ranking documents for a particular search query, ranking algorithms take into account both *on-page* and *off-page* factors. Off-page factors comprise mainly the number and quality of inbound links to a particular page, whilst on-page factors comprise various criteria, most important of which is the relevance of the content to the search query.

3.2 Multi-Document Summarisation

Several different approaches and processes have been developed in automatic MDS systems. These vary according to the problem domain, which usually defines particular formats for both input and output. However, five basic sub-systems of any MDS system can be identified (Mani, 2001).

1. **Unit Identification** During this first phase, input documents are parsed and tokenised into "units", which can vary from single words to whole documents, according to the application problem.

2. **Unit Matching (Clustering)** The second stage involves grouping similar units together. In the context of MDS, similar units usually mean either identical or informationally-equivalent strings (Clarke, 2004), with the purpose of discovering the main themes in the different units and identify the most salient ones.

3. **Unit Filtering** The filtering stage eliminates units residing in clusters which are deemed to be non-salient.

4. **Compacting** During this phase, it is often assumed that different clusters contain similar units. Thus, a sample of units from different clusters is chosen.

5. **Presentation/Summary Generation** The last phase of the MDS process involves using the output from the Compacting stage, and generating a summary. Usually, naïve string concatenation does not produce coherent summaries and thus, techniques such as named entity normalisation and sentence ordering criteria are used at this stage.

3.3 Clustering Techniques

As outlined in Section 3.2, MDS often makes use of clustering techniques in order to group together similar units. Clustering can be defined as a process which performs "unsupervised classification of patterns into groups based on their similarity" (Clarke, 2004).

A particular clustering technique typically consists of three main components:

1. Pattern Representation
2. Similarity Measure
3. Clustering Algorithm

The very generic nature of our problem domain requires a clustering technique which is both suitable and without scenario-dependant parameters. Fung's algorithm (Fung et al., 2003), comprising a pre-processing stage and a further three-phase core process, uses the following concepts, and is briefly described in Figure 1.

ItemSet A set of words occurring together within a document. An ItemSet composed of k words is called a *k-ItemSet*.

Global Support The Global Support of a word item is the number of documents from the document collection it appears in (cf. document frequency).

Cluster Support The Cluster Support of a word item is the number of documents within a cluster it appears in.

1. Pre-Processing - stem, remove stop words and convert to TFxIDF representation
2. Discover Global Frequent ItemSets
3. For each Global Frequent ItemSet (GFI) create a corresponding cluster, containing all documents that contain all items found within the GFI associated with each cluster. This GFI will act as a "label" to the cluster.
4. Make Clusters Disjoint

Figure 1: Hierarchical Document Clustering Using Frequent Itemsets

Frequent ItemSet An ItemSet occurring in a pre-determined minimum portion of a document collection. The pre-defined minimum is referred to as the *Minimum Support*, and is usually determined empirically according to the application.

Global Frequent ItemSet An ItemSet which is frequent within the whole document collection. The words within a Global Frequent ItemSet are referred to as *Global Frequent Items*, whilst the minimum support is referred to as the *Minimum Global Support*.

Cluster Frequent ItemSet An ItemSet which is frequent within a cluster. In this context, the minimum support is referred to as the *Minimum Cluster Support*.

With these definitions, it is now possible to describe into more detail the core non-trivial phases of the algorithm.

3.3.1 Discovering Global Frequent ItemSets

From the definition of an ItemSet, it can be concluded that the set of ItemSets is the power set of all features[1] within the document collection. Given even a small document collection, enumerating all the possible ItemSets and checking which of them are Global Frequent would be intractable. In order to discover Global Frequent ItemSets, the authors recommend the use of the Apriori Candidate Generation algorithm, a data mining algorithm proposed by Agrawal and Srikant (1994). This algorithm defines a way to reduce the number of candidate frequent ItemSets generated. The generation algorithm basically operates on the principle that, given a set of frequent *k-1*-ItemSets, a set of candidate frequent *k*-ItemSets can be generated such that each candidate is composed of frequent *k-1*-ItemSets.

Agrawal and Srikant (1994) also mention a similar algorithm proposed by Mannila et al. (1994). As illustrated in Figure 2, this algorithm consists of first generating candidates, and then pruning the result based on a principle similar to that mentioned.

3.3.2 Making Clusters Disjoint

The purpose of the last phase of the algorithm is converting a fuzzy cluster result to its crisp equivalent. In order to identify the best cluster for a document contained in multiple clusters, the authors define the scoring function illustrated in the equation of Figure 3, where x is a global and cluster-frequent item in doc_j, x' a global frequent but **not** cluster frequent item in doc_j, and $n(x)$ a weighted frequency (TF.IDF) of feature x in doc_j.

Using this function, the best cluster for a particular document is that which maximises the score. In case of a tie, the most specific cluster (having the largest number of labels) is chosen.

4 Procedure

The system was designed in two parts, namely a simple web-based user interface and a server process responsible for iterating sequentially over user queries and performing the content retrieval and summarisation tasks. The following sections describe the various sub-systems that compose the server process.

4.1 Content Retrieval

The Content Retrieval sub-system is responsible for retrieving web documents related to a user's query. This is done simply by querying a search engine and retrieving the top ranked documents[2]. Although throughout the course of this study the system was configured to use only Google as its document source, the number of search engines that can be queried is arbitrary, and the system can

[1] Features here constitute distinct, single words found in the whole document collection. In practice, stemming is applied before feature extraction.

[2] It was empirically determined that retrieving the top 30 ranked documents achieved the best results. Considering less documents meant that, in most scenarios, main relevant sources were missed, whilst considering more documents caused the infiltration of irrelevant information

1. **Join**
   $C_k = \{X \cup X' \mid X, X' \in L_{k-1},\ |X \cap X'| = k-2\}$

2. **Prune**
   $C_k = \{X \in C'_k \mid X \text{ contains } k \text{ members of } L_{k-1}\}$

Figure 2: Candidate Generation Algorithm by Mannila et al. (1994)

$$Score(C_i \leftarrow doc_j) = \sum_x n(x) \times cluster\_support(x) - \sum_{x'} n(x') \times global\_support(x')$$

Figure 3: Definition of Scoring Function

be given a set of parameters to query a particular search engine.

4.2 Content Extraction

The Content Extraction module is responsible for transforming the retrieved HTML documents into raw text. However, a simple de-tagging process is not sufficient. This module was designed so as to be able to identify the main content of a web document, and leave out other clutter such as navigation menus and headings. Finn et al. (2001) introduce a generic method to achieve this, by translating the content extraction problem to an optimisation problem. The authors observe that, essentially, an HTML document consists of two types of elements, that is, actual text and HTML tags. Thus, such a document can easily be encoded as a binary string B, where 0 represents a natural word, whilst 1 represents and HTML tag. Figure 4 shows a typical graphical representation obtained when cumulative HTML tag tokens are graphed against the cumulative number of tokens in a typical HTML document.

Finn et al. (2001) suggest that, typically, the plateau that can be discerned in such a graph contains the actual document content. Therefore, in order to extract the content, the start and end point of the plateau (marked with black boxes in Figure , and referred to hereafter as `i` and `j` respectively) must be identified.

The optimisation problem now becomes maximisation of the number of HTML tags below `i` and above `j`, in parallel with maximisation of the number of natural language words between `i` and `j`. The maximisation formula proposed by the authors is given by Equation 1.

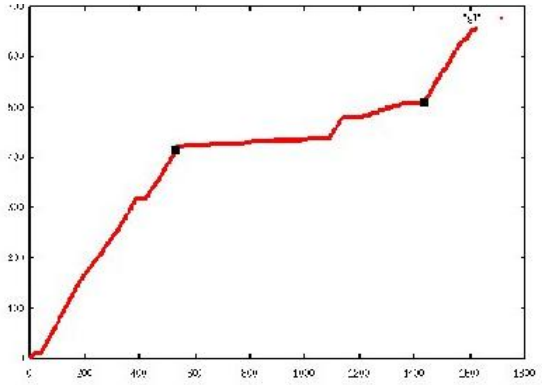

Figure 4: Total HTML Tokens VS Total Tokens (Finn et al., 2001)

$$T_{i,j} = \sum_{n=0}^{i-1} B_n + \sum_{n=i}^{j} (1-B_n) + \sum_{n=j+1}^{N-1} (1-B_n) \quad (1)$$

Our Content Extraction module is further decomposed into three sub-modules. The first is a pre-processing module, which parses out the body of the HTML document, and removes superfluous content such as scripts and styling sections. The second and core sub-module consists namely of an implementation of the content extraction method introduced by Finn et al. (2001), which is primarily responsible for identifying the main content section of the input document. The last post-processing module then ensures that the output from the previous sub-modules is converted to raw text, by performing an HTML detagging processing and also inserting paragraph marks in places where they are explicit cf. HTML `<p>` tag) or where an element from a predefined set of HTML text break delimiters occurs.

4.3 Summarisation

The overall design of the core summarisation module is loosely based upon the two-tiered MDS architecture introduced by Torralbo et al. (2005) The following sections map our system to a similar two-tiered architecture, and explain how each module operates.

Document Identification

Document Identification is trivial, since documents are explicitly defined by the content retrieval module, the output of which is basically a set of query-related text documents.

Document Filtering

The job of Document Filtering is partially done at the very beginning by the search engine. However, our system further refines the document collection by pre-processing each document, applying a noise[3] removal procedure, stemming and stop word and rare word removal. Each document is then converted to a bag of words, or the Vector Space Model, where each word is associated with its corresponding TF•IDF measure. Any document which, after pre-processing, ends up with an empty bag of words, is filtered out from the document collection. Furthermore, in order to ensure the robustness of the system especially in subsequent intensive processing, documents which are longer than 5 times the average document length are truncated.

Paragraph Identification

As outlined in Section 4.2, the Content Extraction sub-system inserts paragraph indicators in the text wherever appropriate. Thus, the paragraph identification phase is trivial, and entails only splitting the content of a document at the indicated positions.

Paragraph Clustering and Filtering

In contrast to the technique of Torralbo et al. (2005), a paragraph filtering module was introduced in order to select only the most informative, query-related paragraphs. To achieve this, we implemented the clustering technique outlined in Section 3.3 in order to obtain clusters of thematically-similar paragraphs, using the Global Frequent ItemSet generation technique from Mannila et al. (1994) and setting the Minimum Global Support and Minimum Cluster Support parameters to 35 and 50 respectively.

[3]"Noise" refers to any character which is not in the English alphabet.

1. For each paragraph p_k
   (a) Initialise the target summary Sum_k as an empty text
   (b) Let $p = p_k$
   (c) Remove the first sentence *s* from *p*, and add it at the end of Sum_k.
   (d) Calculate the similarity between *s* and the first sentence of all the paragraphs, using the size of the intersection of the two vectors of words as a similarity metric.
   (e) Let *p* be the paragraph whose first sentence maximises the similarity, and go back to step (c) with that paragraph. If the best similarity is 0, stop.
2. Choose the longest one of the *k* different summaries.

Figure 5: Summary Generation Algorithm (Torralbo et al., 2005)

The filtering technique then consists of simply choosing the largest cluster. This is based on the intuition that most of the paragraphs having the central theme as their main theme will get clustered together. Therefore, choosing the largest cluster of paragraphs would filter out irrelevant paragraphs. This paragraph filtering method may filter out paragraphs which are actually relevant, however, we rely on the redundancy of information usually found in information obtained from the web. Thus, the paragraph filtering gives more importance to filtering out all the irrelevant paragraphs.

Summary Generation

The role of the summary generation module is to generate a report from a cluster of paragraphs. We based our summary generation method on that used by Torralbo et al. (2005), which is illustrated in Figure 5. However, in order to make it more applicable to our problem domain and increase the output quality, we introduced some improvements.

Sentence Ordering Model We introduced a probabilistic sentence ordering model which enables the algorithm to choose the sentence that

maximises the probability given the previous sentence. The sentence ordering model, based on a method of probabilistic text structuring introduced by Lapata (2003), is trained upon the whole document collection. We used Minipar (Lin, 1993), a dependency-based parser, in order to identify verbs, nouns, verb dependencies and noun dependencies. Using counts of these features and Simple Good-Turing smoothing (Gale and Sampson, 1995), we were able to construct a probabilistic sentence ordering model such that, during summary generation, given the previous sentence, we are able to identify the sentence which is the most likely to occur from the pool of sentences appearing at the beginning of the remaining paragraphs.

Sentence Filtering We also introduced at this stage a method to filter out sentences that decrease the coherency and fluency of the resultant summary. This is based on two criteria:

1. **Very low probability of occurrence**
   If the most likely next-occurring sentence that is chosen and removed from a paragraph still has a very low probabilistic score, it is not added to the output summary.

2. **Heuristics**
   We also introduce a set of heuristics to filter out sentences having a wrong construction or sentences which would not make sense in a given context. These heuristics include:

   (a) Sentences with no verbs
   (b) Sentences starting with an adverb and occurring at a paragraph transition within the summary
   (c) Sentences occurring at a context switch[4] within the summary and starting with a word matched with a select list of words that usually occur as anaphora
   (d) Malformed sentences (including sentences not starting with a capital letter and very short sentences)

5 Evaluation

5.1 Automatic Evaluation

5.1.1 Coherence Evaluation

In order to evaluate the local coherence of the reports generated by the system, we employed an automatic coherence evaluation method introduced by Barzilay and Lapata (2005)[5]. The main objective of this part of the evaluation phase was to determine the effect on output quality when parameters are varied, namely the minimum cluster support parameter for the clustering algorithm, and the key phrase popularity.

From this evaluation, we empirically determined that the optimum minimum cluster support threshold for this application is 50, whilst the quality of the output is directly proportional to the keyword popularity.

5.1.2 Keyword Density Evaluation

Here we focused on determining whether the secondary objective was achieved (cf. section 2).

We measured the frequency of occurrence of the keyword phrase within the output, or more specifically, the keyword density. The average key phrase density achieved by the system was 1.32%, when taking into account (i) the original keyword phrase and its constituent keywords, and (ii) secondary keyword phrases and their constituents.

5.2 Manual Quality Evaluation

In order to measure the quality of the output and determine whether the objectives of the study was achieved, three users were introduced to the system and asked to grade the system, on a scale of 1-5, on several criteria. Table 1 illustrates the results obtained from this evaluation.

6 Conclusions

6.1 Interpretation of Results

In this section we will identify some conclusions elicited from the results obtained from the evaluation phase and illustrated in Section 5.

Automatic Coherence Evaluation The automatic coherence evaluation tests, although, in this application, the level of "coherence" indicated did not match that of manual evaluation, provided nonetheless a standard by which different outputs from the system using different parameters and application scenarios could be compared. From the results, we could empirically determine that the optimal value for the cluster support parameter was around 50%. Furthermore, unsurprisingly, the system tends to produce output of a higher quality

[4] *Context Switch* refers to scenarios where a candidate sentence comes from a different document than that of the last sentence in the summary.

[5] Data required to set up the automatic coherence evaluation model was available from the author's website http://people.csail.mit.edu/regina/coherence/.

	Grammaticality	Non-Redundancy	Referential Clarity	Focus	Structure	Naturalness	Usefulness
Average	3.62	2.21	4.03	4.28	3.27	2.76	4.78

Table 1: Results of Manual Evaluation

in scenarios where the keyword phrase is popular, and thus more data is available.

SEO Evaluation From an SEO perspective, it was predictable that the system would produce query-related text, since its data source is obtained from query-related search engine results. However, the resulting average keyword density achieved is significant, and is at a level which is totally acceptable by most search engine ranking algorithms[6].

Manual Quality Evaluation Due to limited resources, the results of the manual evaluation procedure were not statistically significant since only three users were involved in evaluating six summaries. However, allowing for a factor of subjectivity, some conclusions could still be elicited, namely:

1. The system did not perform well enough to have its output rated as high as a manual summarisation procedure. This can be concluded from the low rating on the output *Naturalness* criterion, as well as from the presence of repeated and irrelevant content in some of the output summaries.

2. The system performed acceptably well in generating reports that were adequately coherent and high-level enough to give an overview of concepts represented by users' queries. This can be concluded from the average scores achieved in the *Focus* and *Referential Clarity* criteria.

3. The evaluators were also asked to give a grade indicating whether this system and similar tools would actually be useful. A positive grade was obtained on this criterion, indicating that the system achieved the MDS objective, enabling users to get a brief overview of the topic as well as facilitating document identification.

When comparing these results to those achieved by Torralbo et al. (2005), we can elicit two main conclusions:

1. Although our system achieved lower rankings on the *Non-Redundancy*, *Structure* and *Grammaticality* criteria, these rankings were not unacceptable. We could attributed this to the more generic domain in which our system operates, where it is not possible to introduce fixed heuristics such as those used by Torralbo et al. (2005) for avoiding repeated information by replacing a term definition by its corresponding acronym. Such heuristics tend to be relevant in the context of such a term definition system.

2. Our system achieved higher grades on the *Referential Clarity* and *Focus* criteria. Given the fact that the system of Torralbo et al. (2005) retrieves results from search engines in a similar way used by our system, the improvement Focus might be attributed to the fact that our paragraph filtering methodology tends to perform well in selecting only the most relevant parts of the document base. Furthermore, the improved grade achieved in the Referential Clarity criterion might arise from the more advanced sentence ordering methodology used, as well as to the different heuristic-based sentence filtering techniques employed by our summary generation module.

6.2 Limitations

The main limitation is that the quality of the output is very susceptible to the quality and amount of resources available. However, we also noticed a severe fall in quality where results were largely composed of business-oriented portals, which tend to lack textual information. Furthermore, the output summary is largely dictated by the results of search engines. Therefore, the queries submitted to the system must be formulated similarly to those submitted to search engines, since the system would fail to generate a focused summary for queries which, when submitted to traditional search engines, return irrelevant results.

The system performance is also limited by the quality and number of external components being referenced, which are not state of the art and which

[6] Very high keyword density (more than a threshold of 2% - 5%) is usually considered as a spammy technique known as *keyword stuffing*.

introduce performance bottlenecks by imposing a batch-processing regime.

6.3 Final Conclusions

Our system combines several existing techniques in a novel way. New techniques, such as our Heuristic-Based Sentence Filtering algorithm, are also introduced.

The primary objective of creating an MDS was achieved albeit with limited "coherency". However, our system was considered a useful research tool - supporting the hypothesis that a partially coherent but understandable report with minimum effort is arguably better than a perfectly coherent one, if the latter is unrealistically laborious to produce.

The secondary SEO objective was also achieved, to the extent that the system generated query-related content that has a natural level of key phrase density. Such content has the potential of being considered query-related also by search engine ranking algorithms, if published within the right context.

7 Future Work

There remains much is to be done. We propose:

- To increase the output quality and naturalness by focusing on an a sub-system for anaphora identification and resolution which would complement our probabilistic sentence ordering model.
- To widen the scope by applying the system to sources of information other than web documents.
- To convert our batch-processing system to an interactive one by incorporating all the required tools within the same environment.

References

Agrawal, R. and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proc. of 20th International Conference on Very Large Data Bases, Sept. 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann.

Barzilay, R. and M. Lapata. 2005. Modeling local coherence: an entity-based approach. In *ACL '05: Proc. 43rd Annual Meeting of the ACL*, pages 141–148, Morristown, NJ, USA. ACL.

Clarke, J. 2004. Clustering techniques for multi-document summarisation. Master's thesis, University of Sheffield.

Edmunds, A. and A. Morris. 2000. The problem of information overload in business organisations: a review of the literature. *Int. Journal of Information Management*, 20(1):17–28.

Eppler, M.J. and J. Mengis. 2004. The Concept of Information Overload: A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines. *The Information Society*, 20(5):325–344.

Evans, D.K., J.L. Klavans, and K.R. McKeown. 2004. Columbia Newsblaster: Multilingual News Summarization on the Web. *Proc. HLT Conference and the NAACL Annual Meeting*.

Finn, A., N. Kushmerick, and B. Smyth. 2001. Fact or fiction: Content classification for digital libraries. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*.

Fung, B.C.M., K. Wang, and M. Ester. 2003. Hierarchical Document Clustering Using Frequent Itemsets. *Proc. of the SIAM International Conference on Data Mining*, 30.

Gale, W.A. and G. Sampson. 1995. Good-Turing Frequency Estimation Without Tears. *Journal of Quantitative Linguistics*, 2(3):217–237.

Google. 2007. *Google Webmaster Guidelines*. http://www.google.com/support/webmasters /bin/answer.py?answer=35769.

Lapata, M. 2003. Probabilistic text structuring: Experiments with sentence ordering. *Proc. of the 41st Meeting of the ACL*, pages 545–552.

Lin, Dekang. 1993. Principle-based parsing without overgeneration. In *Meeting of the ACL*, pages 112–120.

Mani, I. 2001. Automatic Summarization. *Computational Linguistics*, 28(2).

Mannila, Heikki, Hannu Toivonen, and A. Inkeri Verkamo. 1994. Efficient algorithms for discovering association rules. In Fayyad, Usama M. and Ramasamy Uthurusamy, editors, *AAAI Workshop on Knowledge Discovery in Databases (KDD-94)*, pages 181–192, Seattle, Washington. AAAI Press.

Torralbo, R., E. Alfonseca, A. Moreno-Sandoval, and J.M. Guirao. 2005. Automatic generation of term definitions using multidocument summarisation from the Web. In *Proc. Workshop on Crossing Barriers in Text Summarisation Research, RANLP Borovets*.

Vaughn. 2007. *Google Ranking Factors - SEO Checklist*. http://www.vaughns-1-pagers.com/internet/google-ranking-factors.htm.