ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 265-270

An Integrated Neural IR System.

Victoria J. Hodge
Dept. of Computer Science,
University of York, UK
vicky@cs.york.ac.uk

Jim Austin
Dept. of Computer Science,
University of York, UK
austin@cs.york.ac.uk

Abstract. Over the years the amount and range of electronic text stored on the WWW has expanded rapidly, overwhelming both users and tools designed to index and search the information. It is impossible to index the WWW dynamically at query time due to the sheer volume so the index must be pre-compiled and stored in a compact but incremental data structure as the information is ever-changing. Much of the text is unstructured so a data structure must be constructed from such text, storing associations between words and the documents that contain them. The index must be able to index fine-grained word-based associations and also handle more abstract concepts such as synonym groups. A search tool is also required to link to the index and enable the user to pinpoint their required information. We describe such a system we have developed in an integrated hybrid neural architecture and evaluate our system against the benchmark SMART system for retrieval accuracy: *recall* and *precision*.

1 Introduction

Boolean searches approach the Information Retrieval problem at the character string level. They treat queries as symbols linked by logical functions and store the word-document links in a data structure such as an inverted file list [8]. Matching is Boolean, documents either match or they do not, there is no scoring or ranking. Statistical methods approach retrieval at the word unit level, linking words to documents with suitably weighted links. Statistical methods use data structures such as word document matrixes [10] or multi-layer neural networks with weighted links [1]. The documents are scored according to the weight of the links between the query words and the respective documents. The approach suffers the difficulty of selecting suitable word document weight parameters which are critical to retrieval. Statistical approaches have been extended to more knowledge-based approach by Latent Semantic Indexing [2] that aims to extract topic information through word document matrix decomposition. However it is computationally expensive and time-consuming. Knowledge-based approaches include hierarchical document classification [13] or word clustering which may be flat or hierarchical. Document structuring approaches are aimed more at browsing and identifying corpus structure. Flat word clustering such as WEBSOM [7] simply groups words according to their similarities. We generate a hierarchical concept representation of the corpus and can exploit fine-grained or abstract word relationships to calculate document-query similarity and thus rank the documents.

Our overall IR system comprises three modules: a spell checker, a hierarchical thesaurus and finally a word-document association index. We briefly describe each module below with citations linked to more detailed descriptions.

The first module is a **front-end spell-checking system** to isolate any misspelt query words. We use the AURA neural system which provides rapid training and fast partial match retrieval compared to standard neural and statistical approaches [5]. Our spell checker validates each query word against the lexicon using a binary Hamming Distance match. If a word is not matched we assume a spelling error and our spell-checking module provides a list of the best candidate matches for the user to select from. We use a synergistic integration of binary Hamming Distance, n-gram and rule-based phonetic coding approaches to score the candidate matches and generate a ranked list [6]. The user picks the correct spelling. Our spell-checking module also provides a word stemming capability so the user can input a word stem and the spell checker returns a set of stemming variants from the stored lexicon using binary Hamming Distance matching. For example, if the user inputs 'engine' our word stemmer will suggest 'engines', 'engineer', 'engineering' from the lexicon. This allows the user to ascertain all appropriate words in the corpus without requiring a priori knowledge of the corpus vocabulary. Finally, all selected words are awarded an equal score set by the user where a higher score indicates greater importance with respect to the query.

The second module is a **hierarchical thesaurus** we generate automatically from the corpus. We employ a statistical gathering and inference methodology to automatically evolve a hierarchical thesaurus from word co-occurrence statistics in the text corpus. Each word in the corpus is assigned a unique unit length 90-dimensional real-valued vector vector and the seven vectors centred about a specific word are concatenated to produce a context vector representing the central word. All context vectors of a word are averaged to produce a real-valued average context vector. Words are grouped in to clusters by their contextual similarities using the average context vectors and our TreeGCS [4] growing hierarchical clustering algorithm built upon Fritzke's Growing Cell Structures [3]. The clusters form a tree where the cluster contents (word groups) become progressively more specific from root to leaf. We can then exploit the intra-cluster distances and the inter-cluster distances within the synonym hierarchy to ascribe scores to query words and their synonyms. Two clusters that we produced using unstructured text from the Reuters text corpus [9] are:

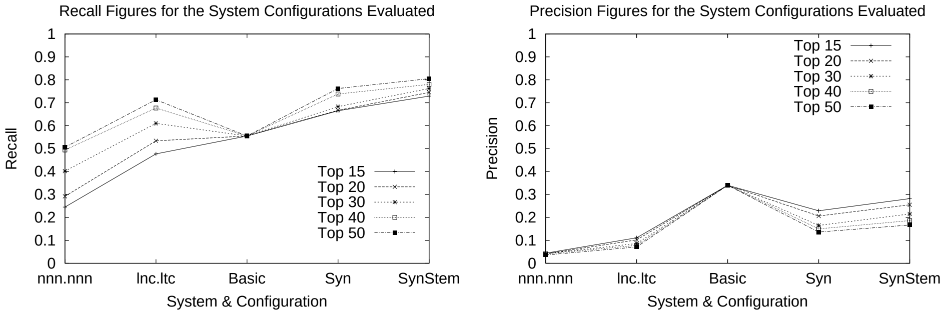
{Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}
{January, February, March, April, May, June, July, August, September, October, November, December, Calendar, Fiscal, Chapter}

The third module is a **binary word-document matrix** using the AURA system for fast training and rapid partial match retrieval [5]. Each row of the matrix effectively indexes a particular word and each column effectively indexes a specific document and a bit is set at position ij if $word_i$ occurs in $document_j$. We index all words except a small set of stop-words (very common words with a very low discriminatory power with respect to text corpora such as 'the' and 'and'). Unlike many other systems, we store word-document associations for infrequent corpus words. We hypothesise that low frequency words may have a low discriminatory power across the corpus but they have a high discriminatory power with respect to the documents that contain them. To retrieve all documents matching a query word, its stemming variants or synonyms, we activate the matrix rows indexed by the words and retrieve all columns where a bit is set. We multiply this output vector by the word score to score the matching documents.

We pass each query word through each of the three modules in turn, retrieving a set of matching documents from the word-document matrix. We produce a separate vector for each query word with an attribute for each document representing the document's score with respect to the specific query word, its synonyms and stemming variants. We take the highest score as the score for each document with respect to that query word. For example, if a document matches the query word 'corn' and its synonym 'maize' awarded 1.0 and 0.5 respectively by the system then the document will take a score of 1.0 for the query word. We can then rank the documents by summing all query word vectors to generate a cumulative document score vector and return the set of matching documents to the user in ranked order.

2 Evaluation

We evaluate our system against the benchmark IR system SMART v11.0 from Cornell University [11]. We evaluate two configurations of the SMART system: 'nnn.nnn' which essentially counts the total number of query words present in each document and ranks the documents according to the word count and 'lnc.ltc' which is the SMART configuration used in the TREC2 evaluation [12] and ranked fourth of the systems evaluated. We evaluate three configurations of our system: 'basic' which retrieves the single set of best matching documents, i.e., the set of documents that matching the greatest number of query words, 'syn' which includes synonymy and 'synStem' which incorporates both synonymy and word stemming. We evaluate all system configurations for recall and precision using 66 queries extracted from the Reuters-21578 Newswire corpus [9]. We extracted the topic sets assigned to the documents in the Reuters



(a) Recall

(b) Precision

Figure 1: Graphs illustrating the recall and precision figures for all system configurations evaluated for the top 15 to top 50 matches retrieved. Our 'Basic' system retrieves a single set of best matches with generally less than 15 documents so we only evaluate the top 15 matches for 'Basic'.

corpus and used these topic sets to generate 66 queries. We determined the correct matching documents for each query; the documents matching the topic set. We read all documents into the respective system configurations and then input each query to each system in turn, retrieving the top 15, top 20, top 30, top 40 and top 50 best matching documents for each query. The *recall* is the number of correct matches retrieved divided by the expected number of matches for all queries. The *precision* is the number of correct matches retrieved divided by the number of matches retrieved for all queries.

3 Results

There are a total of 155 matching documents for all queries as identified by the Reuters' topic assignments for the top 15 matches, 161 for the top 20 and 164 for the other evaluations. We include graphs for the recall and precision in Fig. 1 for easy comparison.

4 Analysis

From Fig. 1, we can see the higher recall and precision figures achieved by our system compared to the SMART configurations for 66 Reuter's queries. There is a significant improvement in the recall and precision figures for our system compared with all SMART variants. We can also observe the improvement in recall and precision when the synonym traversal 'syn' and when the synonym traversal in conjunction with the stemming module 'synStem' is added to the 'basic' system. All system variants (excluding our 'basic' system) have an increasing recall and conversely a decreasing precision as the number of documents retrieved for each query increases from 15 to 50. This is exactly as we would expect as the higher number of retrievals increases the probability of retrieving more correct matches but paradoxically increases the number of false positives if the correct match is not found and thus decreases the precision figure. The highest ranked SMART variant, 'lnc.ltc', has a lower recall figure for the top 50 documents than our 'synStem' configuration achieves by retrieving only the top 15 documents. Our 'synStem' correctly identifies 113 of 155 correct matches in the top 15 with 'lnc.ltc' retrieving 117 of 164 documents in the top 50. For the precision, our system exceeds the highest-ranking SMART precision figure for all evaluations. For the top 50 matches, our 'synStem' variant retrieves 789 documents of which 132 are correct matches, 'lnc.ltc' retrieves 1654 documents of which 117 are correct matches. Our 'synStem' variant retrieves fewer documents in total for the top 50 matches (789 documents in total) than 'lnc.ltc' retrieves in total for the top 20 matches, retrieving 841 documents. We note that we did not select the word stems with any bias towards the dataset, we attempted to select the stem variants we felt most similar to the word stem such as plurals.

Analysing just the top 15 matches. Of the 66 queries, there was 1 query when both SMART configurations found one correct match and all configurations of our system failed to find a single correct match. There were 17 queries when our system 'synStem' found correct matches but 'lnc.ltc' failed, the best performing of the SMART configurations. There were 21 queries where 'synStem' found more correct matches then 'basic' and 2 queries where conversely 'basic' found more correct matches than 'synStem'. For the latter, the 'basic' system found the single correct matches for two queries where the addition of synonym traversal and stemming boosted the scores of other documents and caused the single correct match to fall out of the top 15 matches.

5 Conclusion

We feel the top 15 recall figure of 0.729 and top 50 figure of 0.805 for synonym traversal in conjunction with stemming in our IR system is commendable particularly with the inconsistencies and anomalies of the Reuters' topic assignments with respect to retrieval. The topics were assigned for a train and test classification task and, as such, probably have vagaries and objectivity deliberately included. A 73% success rate for retrieving correct match is very high particularly as this far exceeds the SMART system and our basic system. We feel we have validated the accuracy of our implemented system and demonstrated the necessity of our synonym traversal and stemming modules. We also note that the synonym traversal we employed for this evaluation only clustered 2,192 frequently occurring words of the approximately 49,000 words in the documents. Therefore, we have demonstrated the necessity of such a module and can also surmise that a complete hierarchy, clustering all words in

ESANN'2001 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2001, D-Facto public., ISBN 2-930307-01-3, pp. 265-270

the corpus (minus non-essential words such as stop-words) would improve the recall further.

References

[1] R. Belew, A Connectionist Approach to Conceptual Information Retrieval. In, Procs of the International Conference on AI and Law, 1987.

[2] S. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas & R. A. Harshman. Indexing by Latent Semantic Analysis, Journal of the Society for Information Science, 1(6):391–407, 1990.

[3] B. Fritzke, Growing Cell Structures - a Self-organizing Network for Unsupervised and Supervised Learning. TR-93-026, ICSI, Berkeley, CA, 1993.

[4] V.J. Hodge & J. Austin, Hierarchical Growing Cell Structures: TreeGCS. In, Procs 4th International Conference on Knowledge-Based Intelligent Engineering Systems, 2000.

[5] V.J. Hodge & J. Austin, An Evaluation of Standard Retrieval Algorithms and a Binary Neural Approach. Accepted for publication, Neural Networks: Elsevier Science.

[6] V.J. Hodge & J. Austin, An Evaluation of Standard Spell Checking Algorithms and a Binary Neural Approach. Submitted to, IEEE TKDE.

[7] T. Kohonen, Self-organization of very large document collections: State of the art. In, Procs of ICANN'98, pp 65–74, 1998.

[8] U. Manber & S. Wu, GLIMPSE: A Tool to Search Through Entire File Systems. 1994 Winter USENIX Technical Conference, 1994.

[9] The Reuters-21578, Distribution 1.0 test collection from D. Lewis' home page, currently: http://www.research.att.com/l̃ewis

[10] G. Salton, Interactive Information Retrieval. TR69-40, Cornell University, Computer Science Department, 1969.

[11] SMART, v11 source code: ftp://ftp.cs.cornell.edu/pub/smart

[12] K. Sparck-Jones, Summary Performance Comparisons TREC-2 Through TREC-7. In, Procs of the 7th Text Retrieval Conference: Appendix B, NIST Special Publication 500-242, 1999.

[13] Y. Yang, J. Carbonell et al., Learning approaches for detecting and tracking news events. IEEE Intelligent Systems 14(4):32–43, 1999.