Automated Concept Extraction From Plain Text

Boris Gelfand Marilyn Wulfekuhler William F. Punch III

Genetic **A**lgorithms **R**esearch and **A**pplications **G**roup, the **GARAGe**
Michigan State University, East Lansing MI 48824
{bg, wulfekuh, punch}@cps.msu.edu
http://garage.cps.msu.edu/

Abstract

We describe a system for extracting *concepts* from unstructured text. We do this by identifying relationships between words in the text based on a lexical database and identifying groups of these words which form closely tied conceptual groups. The word relationships are used to create a directed graph, called a Semantic Relationship Graph (SRG). This SRG is a robust representation of the relationships between word senses which can be used to identify the individual concepts which occur in the text. We demonstrate the usefulness of this technique by creating a classifier based on SRGs which is considerably more accurate than a Naïve Bayes text classifier.

1 Introduction

Mining textual information presents challenges over data mining of relational or transaction databases because text lacks any predefined fields, features or standard formats. However, mining information from unstructured textual resources such as email archives, news articles, and the World Wide Web has great potential payoff and has received much recent attention[1][2][3]. One approach for effectively mining relevant information from raw text is based on finding common "themes" or "concepts" in a given document. The field of Information Retrieval is mainly concerned with the specific problem of *search* [4], while we are primarily interested in *concept identification*, which can in turn be applied to a number of specific problems, including search. Some approaches in the Information Retrieval community deal with short *queries*, in which automatic sense disambiguation and query expansion fail to improve search accuracy [5]. Our approach does not suffer from these problems because it uses a much richer starting set, rich enough to allow the system to automatically identify relevant features.

In this paper we describe a method for extracting some semantic features from raw text which are then linked together in a structure which represents the text's thematic content. We call this structure a Semantic Relationship Graph (SRG). No external information about the text in question is required. An SRG relates words according to the appropriate word sense used in the documents, leading to a coherent, cohesive structure which corresponds closely to the idea or ideas in the text.

2 Problem Description

Ours is a bag-of-words approach. We will discount all sense of word order and concentrate on simple occurrences of words. Our goal is to automatically discover some higher level features that

represent some of the underlying semantic concepts which are contained in text. Note that since we are discounting word order, this is clearly a one-way process. In essence, we would like to "distill" a given bag of words into themes much like a human would be able to identify the themes in the same word list. These themes are relatively granular; for our purposes, it would be enough to determine that the sentences "Bill hit Arnold" and "Arnold hit Bill" contain the concepts 'Bill', 'Arnold', and 'hitting', and not answer the question of who actually hit who.

3 Acquiring Semantic Meaning

Suppose a human is given an unordered bag of words which was originally a paragraph of text about a certain complex topic (such as, say, the fall of the Soviet Union) which involves several more basic ideas (such as economic decline, politics, and Soviet Union). A human would be able to identify most of these themes and their interrelationships by just reading the text.

While a human typically has the cultural and linguistic experience to comprehend such a word list, a computer requires a considerable amount of pre–defined knowledge to perform the same task. The knowledge base that we have selected to use is WordNet[1], a lexical database consisting of a number of cross-referenced binary relationships on many nouns and verbs.

WordNet contains information about a number of different types of relationships between words. Currently, we only use sub- and super-class relationships for most common nouns and verbs, but are be able to expand to use other types of relationships found in WordNet [6]. Sub- and super-class relationships are a natural way of relating concepts and provide more organized information about word relationships than a dictionary or synonym-based thesaurus. Also note that WordNet also contains information about the different *senses* of a word. For example, with the aid of WordNet, we are able to tell that the word "club" has four distinct senses: as a suit in cards, as a weapon, as a social group, and as a golf tool.

A key feature of the language relationships we use is that semantically related words tend to be close together, meaning that they are either related directly by a relationship given by WordNet or transitively related through some small group of other words, which we will call these *augmenting* words. In other words, a "simple" or lexical concept (such as economics, from our previous example) is related by the inherent structure of the database used, and not dependent on the exact wording of the original text as long as "some" of the words were there.

Using the WordNet (or a similar) knowledge base, we can construct a Semantic Relationship Graph (SRG) of a word list and attempt to connect each word to other words in the list, adding *augmenting words* not in the original list when necessary for a minimally connected graph. A word is related to another word by either a sub- or super-class relationship if and only if there is a directed edge between the two vertices representing the words in the SRG. The direction of the edge is determined by the type of relationship. It is clear that this model can be expanded to other types of relationships by using different types of links which may correspond to edge weights. This is the subject of current investigation.

3.1 Building the SRG

Building an SRG starts by examining each word in an initial list (called *base* words) and determining which other base words are directly related to it. We then examine all of the words which occur in a single lookup of each base word, and recursively search further and further out along each of the words acquired in successive lookups for the other base words until a threshold depth is reached.

[1] http://www.princeton.edu/~wn

We keep track of the parents of each word, and thus know all the paths from each base word to other base words for every depth. We also keep track of the *sense* of the word and require that all edges coming to or from each word refer to the same sense of that word. Any words that link base words, even if they are not base words themselves, are important *augmenting words* because by linking the base words, they form a part of the underlying concept which would otherwise be incomplete. These augmenting words are then added to the graph at each search depth level, creating a structure which connects as many of the original base words as possible either directly or through a set of augmenting words. Once a certain iteration depth is reached, words not connected to enough other words are thrown out as outliers.

For example, the words "club" and "diamond" may be related by the intermediate word "suit" ("suit" is a generalization of both "club" and "diamond"), and so "suit" would be added to the graph in order to connect "club" and "diamond". If the word "tuxedo" also occurred in the base word set, it may have a connection to the word "suit", but since the senses of "suit" are different, "tuxedo" would not be related in the SRG to "club" or "diamond" via "suit".

The following is an outline of the SRG building algorithm.

Starting with an empty SRG, we do the following for each word:

(1) Set $depth = 0$ and set *wordlookuplist* to be the base word set.

(2) Lookup hypernyms and hyponyms of all words in wordlookuplist, keeping track of the sense and parent of the word.

(3) If we hit another base word add all components of the path (both vertices and edges) which are not already in the graph.

(4) Set *wordlookuplist* to be all of the new words which occurred in the lookup of the hypernyms and hyponyms of the old *wordlookuplist*.

(5) Increment the depth and if below the cutoff, go back to step (2).

We can make several observations at this point. First, note that base words which do not add significant meaning to the text will be thrown out in this process (termed *background* words. This is because they will not connect to other words in the text, as they are not conceptually related. This in essence allows the system to focus on *conceptually* relevant words. We also "fill out" the important or relevant set of words: words that were not in the original text but *should have been* because they were omitted as part of the context of the text. For example, a paragraph which talks about "Senate" and "House of Representatives" may very well never mention the word "Congress" simply because any (English) reader would implicitly understand "Congress" as part of the context of the paragraph.

In this way, we arrive at a structure which relates the words to each other and robustly characterizes the theme of the text that the words were extracted from. Words that were missing from the word list are filled in, words only peripherally related are excluded, appropriate word senses are identified, and we have a relatively complete structure that encompasses and gives order to a conceptualization of the text.

This process is and must be robust since we cannot assume that the initial word set is complete nor can we assume a complete lexical database. There will typically be a number of redundant paths between two connected words which we use to our advantage. If one path is not found due to a missing entry in the lexical database, the system can gracefully compensate for this – it will likely find several other paths to link the words. In fact, words that are linked through only one

path are not likely to be very strongly related. A hierarchical tree approach to this problem, for example, has exactly this weak point; the tree branching is too dependent on the completeness of the knowledge base.

3.2 Example SRGs

Consider the following paragraph, taken from an undergraduate history text[7]:

> Another problem was to make governments strong enough to prevent internal disorder. In pursuit of this goal, however, rulers were frustrated by one of the strongest movements of the eleventh and twelfth centuries: the drive to reform the Church. No government could operate without the participation of the clergy; members of the clergy were better educated, more competent as administrators, and usually more reliable than laymen. Understandably, the kings and the greater feudal lords wanted to control the appointment of bishops and abbots in order to create a corps of capable and loyal public servants. But the reformers wanted a church that was completely independent of secular power, a church that would instruct and admonish rulers rather than serve them. The resulting struggle lasted half a century, from 1075 to 1122.

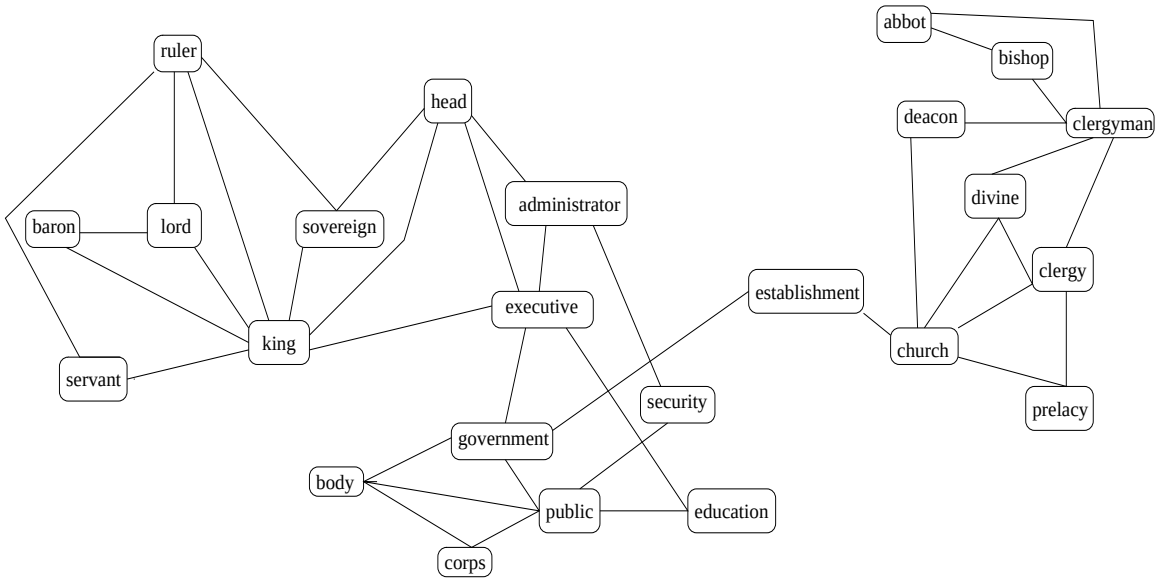

Figure 1: A partially shown SRG of depth 2 when run on the text in the given paragraph. The blocks identified were centered around the government, the Church, and the reform movement(not shown).

We have illustrated the SRG building process by showing the SRG derived from the paragraph of text above. Figure 1 shows an SRG of $depth = 2$ on the actual text words. Notice the strong degree of coherence within the sub-graphs, identifying the lexical-level concepts which make up the theme of the text.

When the same SRG-creating process is performed on a set of random words (from, for example, a unix system's `/usr/dict/words` file), the only connections are sparse and unstructured.

Notice from the example given that the graph tends to be a set of disjoint sub-graphs (or *blocks* which correspond to the simple (lexical) concepts in the text. These lexical concepts come together to form the actual theme or higher-level concept associated with the text. This is the basis for the model which we will refine.

4 Blocking

For the sake of user interaction and concept refinement, it is useful to be able to algorithmically identify the simple lexical concepts which make up the broad theme of the document. In Figure 1, we can notice several distinct well-connected blocks which are in turn loosely connected among themselves.

In an ideal SRG, these blocks would be disjoint sub-graphs, but in many cases we need to deal with "background" connections between words in the SRG. In this case, they consist of the clusters which are more strongly connected among themselves than the background noise connections in the rest of the graph.

To identify these strongly connected subcomponents, it is enough to run a slightly modified single-link clustering algorithm on the square of the adjacency matrix. Squaring the adjacency matrix adds paths of length two in the adjacency lists, with the number of paths between two words also being recorded. This number of paths is a perfect metric for determining how conceptually close together two words are: words that are related by many redundant paths are certainly closer than words related by the occasional link.

This gives us enough information to identify the sub-graphs which are considerably more internally connected than to the rest of the graph, and allows us to extract the individual lexical concepts associated with the text.

5 Experimental results and Applications

There are a large number of applications which can benefit from this type of concept representation, including World Wide Web searching, document summarization, email sorting, or any other application where search and extraction of concepts are more meaningful than mere keywords. To illustrate this in more detail, consider an example web-based classification problem [8]: given a set of pre-classified training pages, in this case web pages of either student or faculty, identify a given web page as either a student or faculty page.

We have implemented a small system to perform the above to verify the strength of our representation technique.

The basic idea is to create a 'master' or 'template' SRG which embodies the semantic ideas in each document class, and then use the master SRGs of each class to determine into which category a new test document falls.

For each class of documents, we create a master SRG which is a union or overlay of all the SRGs of each training document given for each class, keeping track of a relevance count for each vertex. Once these master SRGs are built, we can form a very accurate classifier. Given a test page, we create the corresponding SRG and overlay it upon each master SRG, and identify where it best fits, in terms of vertices unique to only one master SRG and highly relevant vertices which occur more often in one master SRG that any other.

The classification can now take place on a conceptual level based on our understanding of the documents' concepts, rather than solely relying on keyword occurances. Thus, we can determine if a given document *conceptually* fits into a category.

A Naïve Bayes text classifier was implemented to give a reasonable baseline to test our method against[8]. Preliminary results were obtained by creating a master index for each class using from 50 to 400 training examples taken randomly from Mitchell's webkb (http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/) dataset. The same training sets were used to train the Baysean classifier. The test set was 100 documents also chosen at random from the remaining

(non-training) documents. Figure 2 gives a comparison of the performances of the Naïve Bayes and the SRG-based classifiers over varying sizes of the training set.

We are currently in the process of a post-test analysis of the master index SRGs to determine which portions or identifiable *blocks* hold the most discriminatory power. This may lead to a more compact final classifier and thus faster test time.

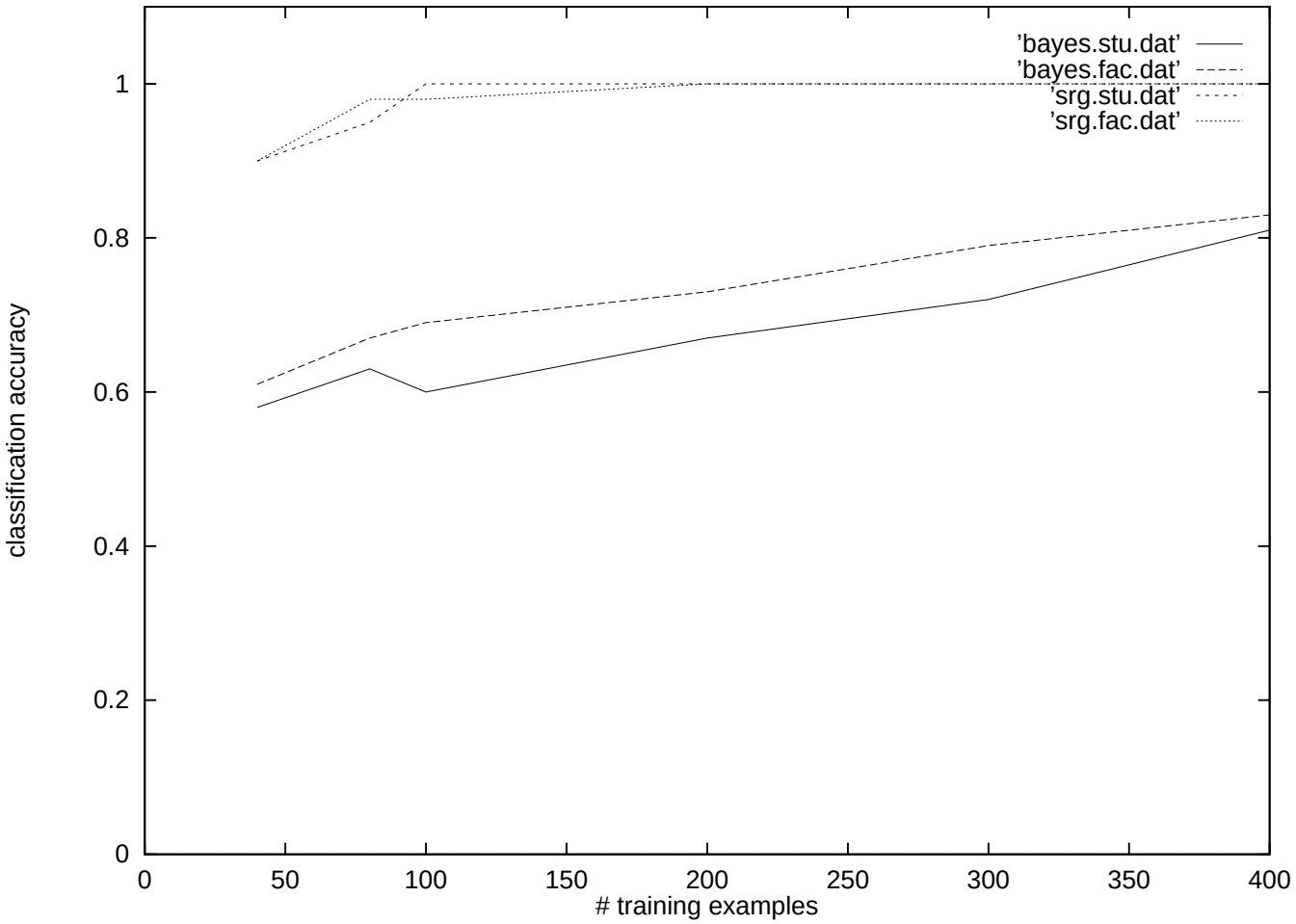

Figure 2: A plot of the relative accuracies of the SRG-based and Naïve Bayes classifiers over the number of examples used per class for training. Each classifier's accuracy for both student and faculty test documents is shown.

5.1 Implementation issues

While the accuracy of the SRG-based classifier is considerably better than that of Naïve Bayes, it should be noted that the run time was also several orders of magnitude higher to create the classifier and push the test set through.

In order to create a production system, several performance issues need to be addressed. In a fully working system, hundreds of SRGs need to be created and analyzed instantaneously, which our current implementation does not allow. A large part of the overhead involved in creating the SRGs are the calls to the database.

Work is currently being done to port WordNet to a fast commercial database. In addition to faster data access, the advantages of this approach include expandability and error checking (enforcing symmetry when appropriate, for example).

Work is also being done to analyze and reduce the algorithmic complexity of the search process, especially in cases with a very large branching factor.

6 Conclusions

We have developed a method to robustly extract and identify structures which closely correspond to concepts from raw text. The resulting SRG directly represents the relationships between the relevant words in the document, either by a direct connection or by the addition of key augmenting words. The relation mapping the SRG space to the true concept space is smooth (similar concepts yield similar SRGs and vice versa), and is the basis for the success of the technique.

If built for a random (concept-less) set of words, the resulting SRG proves to be very disconnected, while an SRG built from a document which indeed contains identifiable (to humans) concepts gives cohesive blocks even at low search depths. SRGs provide a simple, robust mechanism for analyzing unstructured text based on concepts without using sophisticated Natural Language techniques. We have demonstrated the power of this representation when used to classify text documents, and hope to do so in other areas as well.

References

[1] O. Etzioni, "The world–wide web: Quagmire or gold mine?," *Communications of the ACM*, vol. 39, pp. 65–68, November 1996.

[2] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell, "Webwatcher: A learning apprentice for the world wide web," *Proceedings of the 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, March 1995.

[3] M. Marchiori, "The quest for correct information on the web: Hyper search engines," in *Sixth International World Wide Web Conference*, (Santa Clara, CA), April 1997.

[4] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[5] E. M. Voorhees, "Using WordNet to disambiguate word senses for text retrieval," in *Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 171–180, 1993.

[6] G. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, pp. 39–41, Nov 1995.

[7] S. Chodorow, M. Knox, C. Schirokauer, J. R. Strayer, and H. W. Gatzke, *The Mainstream of Civilization Since 1500*. Fort Worth, TX: Harcourt Brace College Publishers, 1994.

[8] C. Y. Quek, "Classification of world wide web documents," Master's thesis, School of Computer Science, Carnegie Mellon University, 1997.