Collections, Collection Exemplars, and the Exemplification Algorithm

Yingwei Wang and Nick Cercone

Department of Computer Science, University of Waterloo
Waterloo, Ontario N2L 3G1 Canada
Email:{ncercone,y25wang}@uwaterloo.ca

1 Collections and Collection Exemplars

We are in the age of an information explosion. People often encounter so many documents that they do not have enough time to read all of them, or even just to scan all of the documents and their titles. Web pages and web sites (represented by their first pages) can be considered as documents. The size and continual growth of the Internet renders this search problem more and more difficult over time.

We attack this problem from a special perspective: getting the general idea of a large document set through some typical documents in the set. First we define collection and collection exemplar:

Definition 1 (Collection) *A document collection (we call it a collection hereafter) is a set of documents which may or may not have an organizational structure superimposed.*

Definition 2 (Collection Exemplar) *Collection C's exemplar is a subset of C that can topically represent the documents in C to some extent.*

Any subset of C can be an exemplar of C. Some exemplars are representative of C. Some exemplars are not representative.

2 Election Analogy

Choosing an exemplar for a collection is not an easy task. How could a subset of a collection represent the whole collection? In what sense is an exemplar qualified as a representative of the collection?

We found that this situation is similar to choosing a government for a country. Choosing a government for a country is also not an easy task and the elected government is often not satisfactory. People have invented many different kinds of election systems to choose a government. In order to get inspiration from the existing election systems, we introduce the following analogy: choosing an exemplar for a collection is like choosing a government for a country.

A country has many people but only a limited number of people can run the government. Whatever election system is used, it is impossible to hear the voices of all people.

A collection has many documents but only a limited number of documents can be part of the exemplar. Whatever exemplification algorithm is used, it is impossible to put all documents into the exemplar.

A government cannot reflect all points of view of the people. But there is a big difference between a good government and a bad government. It is very important to discuss how to choose a good government.

An exemplar cannot reflect all content of the col-

lection. But there is also a big difference between a good exemplar and a bad exemplar. It is also very important to discuss how to choose a good exemplar.

Many aspects of elections need to be discussed, such as riding division, quota of electees, voting options, exemplar choosing, and so on. Due to space limitations, we only discuss two aspects of elections: voting among documents and choosing candidates.

3 Voting Among Documents

We assume that each document is represented by a column vector of word occurrence (Many books and papers, such as [1] and [2], discusses this representation). Using our election analogy, we can imagine that each document is a person, and the words in each vector are the policies with which the person agrees. Suppose in an election person a is a candidate, person b is a voter, we discuss under what condition b will vote for a.

Person b agrees with some policies (contains some words in its vector). For all these policies, he checks if person a also agrees with them. If person a agrees with one policy that person b agrees, person b will be happy and put a Y to denote this. If person a does not agree with one policy that person b agrees, person b will be not happy and put a N to denote this. After person b has checked all the policies with which he agrees, he can calculate b to a's satisfaction rate:

$$satisfaction\ rate = \frac{No.\ of\ Ys}{No.\ of\ Ys + No.\ of\ Ns}$$

If b's satisfaction rate to a is greater than a prescribed value, b will vote for a. Otherwise b will not vote for a.

Now we define b's satisfaction rate to a:

Definition 3 (Satisfaction Rate) *Suppose V_a is the vector of document a, V_b is the vector of document b, $S_b(a)$ is b's satisfaction rate to a.*

$$S_b(a) = (\sum_w sign(V_a[w] * V_b[w]))/(\sum_w sign(V_b[w]);$$

where $sign(x) = 1$ if $x > 0$, $sign(x) = 0$ if $x \leq 0$.

4 Choosing Candidates

Choosing candidates is a very important part of a government election. It is almost impossible for a person who is not a candidate to be elected. Choosing candidates has two major benefits:

The first benefit of choosing candidates is reducing election costs. In an election many costs, like campaign costs, voting costs, statistical costs, are related to the number of candidates. Without specifying candidates everyone in the population can possibly be elected so everyone in the population is a candidate.

Let us imagine that in a government election 1 million people must elect 100 electees. All of the 1 million people are campaigning, each voter has to check all the 1 million candidates to choose his favorite 100 people. How much time and energy would the campaign, the voting, and the statistics cost? Is it not wasteful?

In exemplar elections, election costs are also very crucial. The time complexity of our exemplification algorithm (see Section 5) is $O(k * n)$, where k is the number of candidates and n is the number of documents in the collection. If k is a constant the algorithm is linear. If all the documents in the collection are candidates the time complexity would be $O(n^2)$ so the algorithm will be much slower.

The second benefit of choosing candidates is increasing election quality. Let us consider the example of 1 million people electing 100 electees mentioned. Besides of all the costs, this unreasonable election will also cause the election result to be unsatisfactory. Since each voter has to choose 100 people from a long candidate list of 1 million people and all of the 1 million people are campaigning, the information about all the candidates is too great to be properly processed. It is likely that no common recognition can be reached and the voting result can be very divisive. In this sense we say the election quality is not good.

In exemplar elections choosing candidates can also influence the election quality. If too many can-

didates are involved in a election, voters have too many choices and some less promising candidates may be chosen. If too few candidates are supplied, voter's choices are limited and some promising documents may be missing.

The next big question is how can candidates be determined.

In government elections determining candidates is a complex procedure involving nominations and campaigning and so on.

In exemplar elections we are unable to simulate all these procedures. We have to find promising documents by some heuristic methods.

Candidates should be those documents that are likely to be elected. To find these promising documents we have to make an observation about what kind of documents are likely to be elected. There is no general heuristic method that can be used in all situations.

Here we suggest one heuristic method to find candidates for our exemplification algorithm.

From Definition 3 we know that the satisfaction rate is determined by the number of words in one document covers another document. A reasonable guess is that the more one document has distinct words, the more possible it can cover other documents. This observation leads us to propose the following heuristic method to choose candidates for an election:

Suppose V is the vector of document a. We define the heuristic function $H(a)$ as follows:

$$H(a) = \sum_{w} sign(V[w])$$

where $sign(x) = 1$ if $x > 0$, $sign(x) = 0$ if $x \leq 0$.

If we want to choose m ($m \geq 1$) documents from collection C as candidates, we can calculate $H(a)$ for all $a \in C$, and choose the m documents that have biggest $H(a)$ values.

5 An Exemplification Algorithm

In this section we first describe the algorithm, then we explain some steps of the algorithm.

Algorithm 1 (Exmplification Algorithm) *The inputs of this algorithm are a collection $C[i]$, the maximum number of exemplars $MaxExemplar$, and a prescribed value valve. The output of the algorithm are exemplar documents.*

1. [Init] $n_C \leftarrow$ number of documents in C;
   If $n_C \leq 20$ $MaxCandidate \leftarrow n_C$
   else if $n_C \leq 40$ $MaxCandidate \leftarrow 20$
   else $if n_C \leq 100$ $MaxCandidate \leftarrow n_C/2$
   else $if n_C \leq 500$ $MaxCandidate \leftarrow 50$
   else $if n_C \leq 1000$ $MaxCandidate \leftarrow n_C/10$
   else $MaxCandidate \leftarrow 100$.
2. [Investigate All] For i from 1 to n_C do
   $H[C[i]] \leftarrow$ number of distinct words in $C[i]$.
3. [Choose Candidates] Choose $Can \subseteq C$, where $|Can| \leq MaxCandidate$, $\forall C[i] \epsilon Can$, $\forall C[j] \epsilon C$ but $C[j] \neg \epsilon Can$, $H[C[i]] \geq H[C[j]]$.
4. [Prepare Candidates] For i from 1 to $MaxCandidate$ do
   $VectorCan[Can[i]] \leftarrow$ vector of $Can[i]$.
5. [Vote] For i from 1 to n_C do 6 to 9.
6. [Prepare A Voter] $VectorVoter \leftarrow$ vector of $C[i]$.
7. [One Full Ballot] For j from 1 to $MaxCandidate$ do 8 to 9.
8. [Calculate Satisfaction Rate]
   $S \leftarrow VectorVoter * VectorCan[Can[j]]/H[C[i]]$.
9. [Determine Ballot] If $S > valve$
   $B[i,j] \leftarrow 1$ else $B[i,j] \leftarrow 0$.
10. [Count Ballots] For j from 1 to $MaxCandidate$
    $$Total[j] \leftarrow \sum_{i=1}^{n_C} B[i,j].$$
11. [Prepare Choosing] For i from 1 to n_C do $projection[i] \leftarrow 0$.

12. [Choose All Exemplars] For i from 1 to $MaxExemplar$ do 13 to 16.

13. [Choose One Exemplar] $max \leftarrow 0$;
    For j from 1 to $MaxCandidate$ do 14 to 15.

14. [Modify $Total$ By Overlap Factor] $overlap \leftarrow projection * B[i,j]_{i=1}^{n_C}$;
    $newTotal \leftarrow Total[j]/(overlap + 1)$.

15. [Choose Maximum From All newTotal] If $newTotal > max$
    $max \leftarrow newTotal$; $index \leftarrow j$.

16. [Output The Exemplar] If $newTotal > 1$
    $projection \leftarrow projection + B[i, index]_{i=1}^{n_C}$;
    Output $Can[index]$.

Now we explain some steps of the algorithm:

Step 1: the maximum number of candidates should be calculated from the number of documents in the collection. If n_C is very small, all people can be candidates. If n_C is very big, $MaxCandidate$ has a limit. Between these two extremes $MaxCandidate$ is proportional to n_C and the proportion varies.

Step 2: $H(C[i])$ is the heuristic function that can be used to choose candidates.

Steps 5 to 9 are the voting procedure.

In steps 7 to 9 a ballot of one voter is produced.

Step 8: calculate the satisfaction rate. * is a vector multiplication.

Steps 11 to 16 are used to choose exemplar based on the result of the election. Vector *projection* is used to calculate the overlap among candidate's supporters.

Step 14: * is a vector multiplication.

Step 16: + is a vector addition.

6 Experiments and Results

The exemplification algorithm is designed to find some typical documents in a collection. These chosen documents should be able to represent the collection in some sense.

How can we prove that the chosen documents are typical? We propose the following experiment method:

1. Find some basic collections. We need to ensure that all documents in a basic collection have common subjects.

2. Combine some of these collections together to form a combined collection.

3. Use the exemplification algorithm against the combined collection to get an exemplar.

4. Determine to which basic collection each document in the exemplar belongs. Check if the exemplar can represent the combined collection.

Table 1 describes the basic collections we used in these experiments.

The first column is the collection ID. 8 basic collections are involved in these experiments. The second column is the number of documents in the collection. The third column is the subject of the collection. The fourth column describes the source of the collection.

Table 2 describes the results of these experiments.

The first column is the sequential number of the experiment. There are 19 experiments in total.

The second column describes the composition of the combined collection. For example, in experiment 10 the combined collection was composed of basic collection 2, 3, and 4 described in Table 1.

The third column is the number of documents in the combined collection. Among these experiments, the smallest combined collection contains 7 documents, and the largest combined collection contains 2007 documents.

Table 1: Basic Collections

Collection ID	Size	Subject	Source
1	31	Latex	Software Documentation
2	4	Flashpoint	Software Documentation
3	3	Aladdin	Software Documentation
4	20	University of Waterloo	Internet
5	316	Document Collection	Internet
6	251	University of Waterloo	Internet
7	622	Oktoberfest	Internet
8	818	Christmas	Internet

Table 2: Experiment Results

Experiment	Composition	No. of Docs	Result	First	Second	All
1	{1,2}	35	1	Y	N	N
2	{1,3}	34	1	Y	N	N
3	{1,4}	51	1,4,4	Y	Y	Y
4	{2,3}	7	2,2,3,3	Y	Y	Y
5	{2,4}	24	4,2,4,4,4	Y	Y	Y
6	{3,4}	23	4,3,4,4,4	Y	Y	Y
7	{1,2,3}	38	1	Y	N	N
8	{1,2,4}	55	1,4,4	Y	Y	N
9	{1,3,4}	54	1,4,4	Y	Y	N
10	{2,3,4}	27	4,2,2,4,4	Y	Y	N
11	{1,2,3,4}	58	1,4,4,4,4	Y	Y	N
12	{5,6}	567	5,5,5,5,5	Y	N	N
13	{5,6,7}	1189	7,5,5,7,5	Y	Y	N
14	{5,7}	938	7,5,5,7,7	Y	Y	Y
15	{6,7}	873	7,6,7,6,7	Y	Y	Y
16	{7,8}	1440	8,7,7,8,7	Y	Y	Y
17	{5,7,8}	1756	8,7,5,5,5	Y	Y	Y
18	{6,7,8}	1691	8,6,7,7,8	Y	Y	Y
19	{5,6,7,8}	2007	8,6,7,5,5	Y	Y	Y

The fourth column describes the result of this experiment. For convenience, we give the basic collection IDs corresponding to the documents in the exemplar instead of the exemplar document names. For example, in experiment 8 the exemplar contains three documents, the first document of the exemplar is from basic collection 1, the second and third documents of the exemplar are from basic collection 4.

The fifth column answers this question: does the first document in the exemplar belong to the biggest basic collection? For example, in experiment 16 the first document in the exemplar belongs to basic collection 8 which is bigger than all the other basic collections in this composition.

The sixth column answers this question: does the exemplar cover the first two biggest basic collections? For example, in experiment 18 the exemplar covers the first two biggest basic collections (8 and 7), so this column is Y.

The last column answers this question: does the exemplar cover all basic collections in this combined collection? For example, in experiment 19 the exemplar covers all 4 basic collections (5,6,7,and 8) in the combined collection, so this column is Y.

From Table 2 we know that the exemplification algorithm

1. can always find a document to represent the biggest basic collection;

2. in most cases can find a document to represent the second biggest basic collection;

3. in about half the cases can find an exemplar that covers all the basic collections in the composition.

These experimental results show that an exemplar produced by the exemplification algorithm can represent the collection to some extent. We are doing further research to improve and evaluate this algorithm.

References

[1] Willian B. Frakes and Ricardo Baeza-Yates. *Information Retrieval, Data Structures and Algorithms.* Pretice Hall, Englewood Cliffs, New Jersey, 1992.

[2] Daniel Boley. Principal direction divisive partitioning. Technical Report TR-97-056, Department of Computer Science, University of Minnesota, Minneapolis, 1997.