Class Evaluation Summarizer

Arun Ramachandran, Swaminathan Sivaraman, and **Sriram Sundar**
Final Course Project (Application), CSE 537 - Artificial Intelligence
Stony Brook University
Stony Brook, New York 11794
{aruramachand,ssivaraman,ssundar}@cs.stonybrook.edu

Abstract

Manual summarization of student course feedback involves a lot of time, money and energy. In here, we seek to build an application, that automatically performs extractive summarization, based on an integer linear programming (ILP) framework. The approach also builds correlation between student responses based on latent semantic analysis, and avoids sparsity problems. We show that our application performs better than the ILP-baseline system, when bigrams are chosen as concepts. We also list out areas where the application fails to perform better than the ILP-baseline, and indicate potential reasons for it.

Introduction

Problem Definition

This project deals with developing an application to perform extractive text summarization of student course feedback. Extractive summarization identifies important sentences from a document, under a given length constraint. Given a document or a set of documents containing feedback of each student about a course/courses, this application will generate the summary of the feedback per course. Students are generally asked to evaluate the teaching faculty during the end of the semester, in which each student is asked to provide comments about what parts of the class were interesting, what parts were unclear, and what were the important learning outcomes. An example set of student responses for a question about what parts of the course they found interesting and a human-evaluated summary of the responses is presented in Table 1. What our application does is to extract the best set of sentences from the response that summarize what was interesting in the class (extractive summarization). It has to be noted that these would not be the same as the reference summaries listed above as that has more intelligence incorporated into it. The goal is to get our summaries as close to the reference summary as possible.

Student Responses:
Learning how conduction works with energy levels
Breaking down the motorcycle into parts for analysis
How diamond is a very good thermal conductor
Learning the different properties of bonding
How bonding greatly affects the properties of materials
Diamond thermal conduction
The interesting point was about the thermal properties of diamonds
The differences between bonding and how they relate to the kinds of materials they produce
Human-evaluated Summary:
Different bonds and their effects on thermal properties
Diamond thermal conduction
Motorcycles parts activity

Table 1: A table of student responses and a human-evaluated summary to the question "What parts of the course did you find interesting?" in a science class.

Motivation

Earlier, either teaching faculty or staff designated for summarizing course feedback had to manually look at all the summaries and determine the crux of the learnings and concerns of students as a whole. For a small class, this is easier. But for a large environment, this becomes cumbersome, difficult to scale and inconsistent across years (due to possible involvement of different people). Our application would alleviate this trouble, and provide a concise summary of the course feedback across a wide range of topics. This would make it easier to scale, reduce human involvement (subjectivity), and the time and energy saved could be used for further analysis.

Contributions

There have been supervised approaches for this problem with classifiers to predict summary sentences. But we have developed this application using an unsupervised approach, based on the Integer Linear Programming Framework (ILP) developed by Gillick and Favre (Gillick and Favre, 2009). This application also uses a latent semantic analysis technique to identify semantically important sentences, and remove concerns over sparsity of concepts in sentences. This algorithm is called SOFT-IMPUTE, developed by Mazumdar et al.(2010). This is important because different words or their combination could be used to express the same mean-

ing, like "How bonding greatly affects the properties of materials" and How bonding relates to the kinds of materials they produce". We need to correlate them in order to generate better summaries. In order to evaluate our application which has the latent semantic analysis incorporated, we compared it with the baseline ILP model which we implemented. The evaluation was done using the ROUGE summary evaluation package, which is a widely acknowledged standard for evaluating text summarization. We then analyzed the results, and determined where it worked and where it failed. We also indicated potential areas of research and exploration, so that this application could become invaluable for teaching faculty.

Learning Outcomes

We learned new techniques and algorithms. The first was how to model a problem as an ILP problem, and how to implement the solution and evaluate it. The next was an algorithm called SOFT-IMPUTE, which makes sparse co-occurrence matrices denser. Co-occurrence matrix is a matrix of mappings from term/concepts to documents/sentences. Terms could be a combination of n-grams with different features. The SOFT-IMPUTE algorithm is based on the Singular Value Decomposition (SVD) technique, which is widely used to make a sparse co-occurrence matrix denser, by determining associations between the terms/concepts. We also learnt to use the ROUGE evaluation package to evaluate text summaries.

Our application would work to produce representative summaries on an input document containing sentences with student response, one response per line. It works better when we use bigrams as terms/concepts, than any other combination, and especially when stop-words were removed. The recall-value, as is generally observed with text summarization, was lesser when we evaluated our application.

Description

Two approaches have been explored to solve this problem - one is a Integer Linear Programming model-based approach and the other is a Soft-Impute approach that builds on the ILP formulation.

ILP Approach

This approach formulates the problem as a Linear Programming problem and tries to maximize an objective function under certain constraints. Let S be the total set of student responses, that has totally M sentences. The final summary will be a subset of these sentences. We split each sentence into bigrams (or other variants) and get the total weight of each bigram in the document. Let N be the total number of unique bigrams and w_i denote the weight of each bigram (i goes from 0 to $N-1$). We will choose which of these bigrams are important enough to be included in the summary. Let $y_j \in \{0,1\}$ (j goes from 0 to $M-1$) indicate if sentence j is selected in the summary ($y_j = 1$) or not ($y_j = 0$). Similarly, let $z_i \in \{0,1\}$ (i goes from 0 to $N-1$) indicate if bigram i is important ($z_i = 1$) or not ($z_i = 0$). We will construct a co-occurrence matrix A of dimensions NxM where the rows represent the bigrams and the columns, the sentences. $A_{ij} = 1$ denotes that bigram i occurs in sentence j and $A_{ij} = 0$ denotes that it doesn't. If we assume that we limit the overall summary character length by L, we need to find sentences that have maximum overlap with high-weight bigrams. So, we can formulate the objective function as,

$$\max_{y,z} \sum_{i=0}^{N-1} w_i z_i$$

and the constraints as,

$$\Sigma_{j=0}^{M-1} A_{ij} y_j \geq z_i$$

$$A_{ij} y_j \leq z_i$$

$$\Sigma_{j=0}^{M-1} l_j y_j \leq L$$

where l_j is the character length of sentence j and any z_i or y_j can only be in $\{0,1\}$. After solving this, we will have a list of important sentences and the important bigrams in them. We will directly return the full important sentences (inspite of possibly unimportant bigrams in them) as we want to return coherent results.

SOFT-IMPUTE approach

SVD The problem with the ILP approach, is that the co-occurrence matrix is sparse. This problem is dubbed as the matrix completion problem(Candès and Recht, 2008; Candès and Tao, 2009). Generally, less than 5% is likely to be filled with non-zero values. This leads to a lack of sufficient information about semantically similar concepts(bigrams) across sentences. There could be bigrams like "not comprehend" and "lacked clarity", which are just different ways of expressing the same meaning. We need to distribute the importance scores for them among both, even if one of them appears in a sentence. Yihong Gong and Xin Liu had published the usage of Latent Semantic Analysis technique in text summarization, by applying a singular value decomposition to the co-occurrence matrix. The SVD of the co-occurrence matrix A of dimensions NxM, is defined as:

$$A = U \Sigma V^T$$

, where $U = u_{ij}$ is an NxN column-orthonormal matrix, whose columns are the left singular vectors(which are mutually orthogonal (i.e.) $u_m.u_n = 0$, and are normal (i.e.) $\|\mathrm{u}_m\| = \|\mathrm{u}_n\| = 1$); $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_{min(m,n)})$ is an NxM diagonal matrix, with diagonal elements being non-negative singular values in decreasing order, and $V = v_{ij}$ is an MxM column-orthonormal matrix, whose columns are the right singular vectors. If rank of the matrix A is r, then

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_{min(m,n)} = 0$$

What it means is that, the SVD derives the latent semantic structure of the matrix A by breaking it down into r linearly independent base vectors/concepts/bigrams(this means that other vectors/concepts/bigrams can be represented using these r base vectors/concepts/bigrams itself). A critical feature of SVD is that it can map correlation among bigrams, so as to cluster bigrams and sentences, and indicate which

bigram is recurring and hence important. Thus, each singular vector of U represents an important bigram/concept, whose corresponding singular value represents the amount of importance it has. Similarly, from the singular vectors of V, we can determine the most important sentence for each bigram.

SOFT-IMPUTE Algorithm Though various algorithms have been in use to complete a sparse co-occurrence matrix, they have suffered from limitations like not having a regularization parameter or not being scalable with matrices of large dimensions. So, we used the SOFT-IMPUTE algorithm in our application, to complete the missing values in the high-dimensional sparse matrix, A, by approximating it using a low-rank matrix. We seek to find a low rank matrix B, with the same dimensions as A, which is as close to A as possible, at the observed positions (now since our data set is small, with on an average 150 bigrams and 50 sentences leading to 7500 positions, our matrix is generally fully observed. However, this SOFT-IMPUTE algorithm would also work for large-scale problems in which we can't observe all positions). The objective function can be represented as

$$\min_B \frac{1}{2} \sum_{(i,j)\in\Omega} (A_{ij} - B_{ij})^2 + \lambda \|B\|_*$$

where Ω is the set of observed positions, $\|\mathrm{B}\|_*$ is the trace/nuclear norm of B,

$$\|B\|_* = \sum_{i=1}^{r} \sigma_i$$

r is the rank of matrix B, and σ_i are its singular values. $\lambda >= 0$ is the regularization parameter controlling the trace norm of the minimizer $\hat{B}_\lambda$. At every step, the algorithm decreases the value of the objective function towards its minimum, and gets closer to the optimal completion of the sparse matrix A. At each step, the relatively less sparse matrix can be split as follows :

$$B = B_{sparse} + B_{lowrank}$$

B_{sparse} has the same sparsity as A, and $B_{lowrank}$ has rank $\hat{r} << M$, N and $\hat{r}$ is very closer to r, which is the rank of matrix B when the algorithm converges. This leads to A_{ij} assuming values in the continuous range [0,1], a shift from earlier when it assumed only discrete values (0 or 1).The bigram variables, z_i can also assume continuous values in the range [0,1]. Based on Cai et al.(2008), we define a projection matrix P_Ω as

$$(P_\Omega(X))_{ij} = \begin{cases} X_{ij}, & \text{if } (i,j) \in \Omega. \\ 0, & \text{otherwise.} \end{cases}$$

So, our objective function becomes

$$\min_B \frac{1}{2} \sum_{(i,j)\in\Omega} (P_\Omega(A) - P_\Omega(B))^2 + \lambda \|B\|_*$$

which is the same as

$$\min_B \frac{1}{2} \|(P_\Omega(A) - P_\Omega(B))\|_F^2 + \lambda \|B\|_*$$

Feedback Question
Q. What were the learning outcomes of today's lecture?
Human-produced Reference Summary
- Visual examples and graphs
- Group activities
- Real life examples
Our Summary - ILP with Baseline and Bigrams
- Examples helps a lot
- Need bigger pictures
- Reviewing HW helps
- I like getting the real world examples
- Activities, particulary ones with drawing visuals help a lot
- Visual aids are helpful
Our Summary - Soft-Impute with ILP and Bigrams
- Images and visuals help understand content
- Activities, particulary ones with drawing visuals help a lot

Table 2: A sample of human-produced reference summary and summary generated by our system.

where $\|.\|_F$ is the Frobenius norm, the square root of the sum of squares of the differences of corresponding terms. Based on Mazumdar et al.(2010), the solution to the above function is given by

$$\hat{B} = \mathrm{SVD}_\lambda(A) = U \Sigma_\lambda V^T$$

where $\Sigma_\lambda = \mathrm{diag}[(\sigma_1 - \lambda)+, ..., (\sigma_r - \lambda)+]$,
SVD(A) = U ΣV^T,
$\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_r)$,
and $t+$ = max(t,0).

The notation $\mathrm{SVD}_\lambda(\mathrm{A})$ is called soft-thresholding. Thus, we iterate for λ_1, λ_2,... upto λ_k for k iterations or till the convergence threshold is reached. The final $\hat{B}_{\lambda_k}$ is our final dense matrix. The algorithm is the Algorithm 1 SOFT-IMPUTE of Mazumdar et al.(2010). The convergence rate of the algorithm is O(1/k), where k is the number of iterations.

Evaluation

Dataset

The dataset we use includes students' responses collected from a science class taught in a university. The dataset is publicly available[1]. The feedback is taken from students in 3 separate categories after each class. The questions are segregated into three categories as follows

1. The most interesting part of the day's lecture, i.e. the parts that interested the student and made him/her more involved in the lecture.
2. The worst part of the day's lecture i.e. the parts which were not clear to the student.
3. The learning outcome of the lecture, i.e. what was the take away from the lecture.

[1] www.coursemirror.com/download/dataset/

Configuration	ROUGE-1			ROUGE-2			ROUGE-3			ROUGE-SU*		
	R	P	F	R	P	F	R	P	F	R	P	F
ILP + Bigram	34.1	30.3	29.1	10.2	10.1	9.4	4.6	3.9	3.9	10.9	9.4	7.3
Soft-Impute + Bigram	34.6	37.6	32.5	14.8	15.7	14.1	6.1	6.4	5.8	11.9	15.4	9.8
ILP + Stopwords Included + Bigram	37.1	28.8	29.5	10.6	10.3	9.8	4.9	4.4	4.4	12.5	8.8	7.3
Soft-Impute + Stopwords Included + Bigram	25.8	25.7	22.5	3.5	5.3	4.1	1.3	1.8	1.5	7.01	8.6	5.0
ILP + Unigram	38.8	17.9	22.3	7.3	2.9	3.9	0.7	0.2	0.3	13.9	3.3	3.8
Soft-Impute + Unigram	18.0	20.3	16.5	4.0	4.0	3.7	1.1	1.3	1.2	4.81	6.3	3.6
ILP + {Uni,Bi}gram	40.4	18.4	23.2	7.0	2.9	3.9	0.1	0.1	0.1	15.1	3.5	4.2
Soft-Impute + {Uni,Bi}gram	29.7	19.2	21.1	0.3	0.2	0.2	0.0	0.0	0.0	11.3	4.9	5.1

Table 3: Averaged ROUGE scores for similarity between the gold standard and our output for ILP and Soft-Impute approaches with different configurations. 'R' is Recall, 'P' is precision and 'F' is the F-score. All scores are in percentages.

So, each of the students is provided with this questionnaire after each lecture. All these responses for a day are concatenated and put into a single document. The average length of each response is approximately 10-11 words.

These are the test datasets used for our evaluation. Our model generates summaries for each of the above three prompts for every lecture based on responses collected from all 53 students in the class.

The reference summaries that we use to test our generated summaries are those provided by the Teaching Assistant (TA) after each class. Each document in the dataset consists of feedback from 53 students as well as the consolidated summary given by the TA. On average the summary of TA (reference summary).

Evaluation Metric

We use ROUGE (Recall-Oriented Understudy of Gisting Evaluation) (Lin, 2004) to evaluate our summaries. ROUGE is a software package for automated evaluation of summaries. We use Pyrouge, a python wrapper for ROUGE package which converts our generated summary and the gold standard summary (the reference summary of the TA) into a format that ROUGE understands.

We have implemented a few variants of the baseline configuration. By default, stop-words are removed (unless otherwise stated). The various configurations we experimented with are as follows,

- ILP and Soft-Impute with unigrams
- ILP and Soft-Impute with bigrams
- ILP and Soft-Impute with unigrams + bigrams
- ILP and Soft-Impute with bigrams and stop-words included

The system that Pyrouge compares our result is a set of references (human-produced) summary. The results obtained are described in the next section.

Results and Explanations

Table 2 presents the reference summary and the summary generated by our system for a given feedback question during a particular class. We evaluate all such sample summaries generated by our system with the reference summaries using ROUGE. Table 3 depicts the averaged Recall, Precision and F-scores of various configurations of our system.

We first tried out the ILP and Soft-Impute version with only bigrams. The ILP+Bigrams version performed reasonably well as a baseline. We re-ran the tests with Soft-Impute+Bigrams and noticed an increase in all three scores - Precision, Recall and F-score, showing that the Soft-Impute algorithm does provide an increase in the quality of the summary when compared to the ILP version.

Alternate configurations We then tried the same experiment with 3 other different configurations - bigrams with stopwords included, unigrams and unigrams + bigrams. We observed that in all 3 cases, the Soft-Impute scores are poorer than the ILP scores. None of the scores beat the Soft-Impute+Bigram version however. Hence, it might be argued that the Soft-Impute algorithm is most effective when used with bigrams. Also, since no other configuration could beat this version, this configuration seems best suited for this application. The unigram models possibly aren't working well since it is widely accepted that bigrams have performed better than unigrams in NLP applications. Addition of stopwords is obviously lowering the score since it is just redundant information and is messing up the bigram frequencies.

Limitations and Future Scope

Given a course feedback text corpus, our system selects the most important set of sentences required to represent the entire text succinctly and builds a summary. However, we do not extract relationships present between sentences and hence do not identify similar sentences. Thus, there are possibilities of two sentences quite similar in meaning to be selected as important sentences and occur in our summary. In the future, we can improve the system to resolve such sentences which refer to the same meaning or context.

One other major limitation is that the current system does not handle negative phrases well. For example, if many students wrote "did not grasp the concepts well" and not as many wrote "was able to grasp the concepts well", there is a high chance that the final result will have the second statement, while it is the opposite of the truth! A possible solution is to do POS-tagging of each sentence, identify negation words like 'not' etc., identify the closest verb or adjective to its right and remove the bigrams after that word from our

dataset. In this case, we would remove 'grasp concepts' and 'concepts well' ('the' is a stopword, so ignored) for the negative cases. But this would still not handle quasi-negative phrases like 'hardly' or 'rarely' and double-negative phrases. Possibly, we can add features and use models to detect the presence of such phrases or even ambiguous or sarcastic sentences and rephrase them. This would provide the users of our system a more precise and realistic summary of the text.

Conclusion

Our student course feedback summarization application finds a representative subset of sentences from a text corpus containing information of the entire set of sentences. We obtain a subset of the most important sentences using an integer linear programming(ILP) framework and experiment with different variants of the system to determine the best possible configuration for our system. When using a combination of ILP, Soft-Imputing, stopword filtering, and bigrams to summarize the course feedback, we soft-imputed the data and successfully managed to improve the ROUGE score by a fractional margin. We also used the low-rank approximation of the co-occurrence matrix to represent relations between bigrams and their respective occurrences in sentences present in the corpus. This enabled us to overcome matrix sparsity issues. The evaluation of our system with baseline configurations showed that our application (ILP + Soft-Imputing + Bigrams + Stopwords removal) performs marginally better than the baseline configuration at most times. This encourages further exploration and shows promise to develop more sophisticated feedback analysis over summarized text in future.

References

Luo, W., Liu, F., Liu, Z. and Litman, D. 2016. Automatic Summarization of Student Course Feedback. In *Proceedings of NAACL*, 80-85.

Gillick, D. and Favre, B. 2009. A Scalable Global Model for Summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*, 10-18.

Mazumdar, R., Hastie, T. and Tibshirani, R. 2010. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287-2322.

Steinberger, J. and Jezek, K. 2004. Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. *Proceedings of ISIM 2004, Roznov pod Radhostem, Czech Republic, April 2004*, pp.93-100

Candès, E.J. and Recht, B. 2008. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9:717-772.

Candès, E.J. and Tao, T. 2009. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053-2080.

Cai, J., Candès, E.J, and Shen, Z. 2008. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956-1982.

Lin, C.Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the Workshop on Text Summarization Branches Out*.