CSE 592 - Project Report
Hierarchical Co-clustering of Comic Series and their Instances

Chandra Sekhar Mallarapu
Naresh Pratap Singh
Nehal Bandi
mcsekhar,mail2naresh,nehal.bandi@gmail.com

December 22, 2011

Contents

List of Figures

1 Abstract

In this Internet age, the number of comics available on the web have exploded. Avid comic readers do not have an easy way to find comics similar to what he/she is interested in. This motivates us to solve the problem of grouping together comics which are similar. We intend to achieve two goals. First, we should be able to tell user which comics in the same series are similar. Second, we should also be able to suggest new interesting comics.

2 Introduction

People have been using the medium of comics since a very long time. Comics were traditionally found only in newspapers and print material, for which people had to pay money. But in the present day, a number of people have started their own web based comic series. Another interesting fact is that some of the comics which are printed in newspapers are also available online. Hence, readers are looking more and more to the web for their comic needs. A user who is reading some comic, is naturally interested in finding out other similar comics. We decided to solve this problem by creating groups of similar comics together, as that will aid in future navigation and search.

At a first glance, one might think of using image analysis for this. That approach might work with mixed results, if one only wanted similar comics within a series. But, it is neither easy nor promising when attempting to find similarities across various series. Because, images in comics are not standard across series, and are usually very simple and do not represent real world entities faithfully. Another approach is to mine blogs and use search engine results to get information about similarities of comics.

Our approach is to use the information in the comic transcripts to find similarities. This is a suitable approach, because in most cases, the transcript contains not just the text of the comics, but also additional details describing the image, scene and setting too. Moreover, one doesn't have to do complex image analysis to understand what the comic is about. And in our opinion, its an easy and better way to understand and analyse the comic. Thus, the transcripts constitute our data samples. We can now do clustering on this data to find out groups of comics similar to each other. We can also use the comic series themselves as samples, and cluster them. Moreover, we decided to go one step ahead and do a hierarchical clustering, as that will allow to create coarse grained clusters too. Since, similarity about comic instances hints at similarity between the comic series they belong to and vice versa, we felt that simultaneously clustering series and instances, also called

co-clustering, is a good way to solve the problem described earlier. Thus, the comic series and individual instances from all series constitute our data samples. The features needed will have to be generated from the transcripts of the comics. As we have those transcripts we can use natural language processing techniques to find relevant features and similarity measures. Using that information clusters can be created. The clusters would be useful to find similar comics, as well as browsing comics in general.

Eckes et al, [2] have described an algorithm to do hierarchical co-clustering. A modification of this called *HCC* is proposed in Jingxuan Li et al, [1]. We have modified this and used it in our project.

3 Dataset

To collect data for our project, we have written a crawler, which downloads the HTML from a web page. We pass it a list of comics series to download. The crawler then searches the web for all the available comic transcripts for that series. It parses them and stores on the disk. This way, we collected transcripts of a few popular comics. The size and distribution of our dataset is as follows:

- About 20528 documents from 9 comic series
- Downloaded following comic series from OhNoRobot[3]
  - Calvin And Hobbes - 3696 instances
  - College Roomies From Hell - 2631 instances
  - Diesel Sweeties - 1778 instances
  - Goats - 2266 instances
  - General Protection Fault - 3298 instances
  - Nukees - 889 instances
  - Questionable Content - 1747 instances
  - Sheldon - 3263 instances
- Xkcd - 960 instances
  Available from XKCD [4]
- Calvin and Hobbes from GoComics[5]

4 Hierarchical Co-clustering

We use hierarchial co-clustering which utilizes mutual information between the series and the comics. Document clustering provides additional information for series clustering while series clustering supplements document clustering. Given two types of data objects $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ and a matrix X, such that $x_{ij} \in \mathbb{R}$ represents the relation between a_i and b_j. We need to simultaneously cluster objects in A and B, and create hierarchies out of those clusters. Here we describe a modification of the *HCC* algorithm presented in [1]. We start off with singleton clusters containing a tuple, which is a pair of a single document and a single series. We then combine two nearest clusters in each iteration, until there is only one cluster. Thus, a dendrogram is generated, with the leaves being tuples containing a single document and a single series, and internal nodes containing a cluster of such tuples. At the root, we have all such tuples. The modified *HCC* algorithm is described below.

Algorithm 1 HCC Algorithm Description

Create an empty hierarchy H
$List \leftarrow$ Objects in $A\ X\ B$
$N \leftarrow |A\ X\ B|$
for $i = 0$ to $N - 1$ **do**
  p, q=PickUpTwoNodes($List$)
  o=Merge(p, q)
  Remove p, q from $List$ and add o to $List$
  Add $List$ to H as next layer
end for

The main step in this algorithm is to find out which two nodes have to be merged, which is what the *PickUpTwoNodes* procedure does. It works as given below. Given a cluster C with m tuples, the cluster heterogeneity of this cluster $CH(C)$ is defined as:

$$CH(C) = \frac{1}{|C|} \sum_{(a_i, b_j) \in C} (x_{ij} - \mu)^2$$

where $\mu = \frac{1}{|C|} \sum_{(a_i, b_j) \in C} x_{ij}$. Another choice for μ was given in [2] in which they used $\mu = max(x_{ij}), \forall x_{ij} \in X$. We ran our experiment using this measure too.
The clusters to be merged are found out by calculating $CH(C_1 \cup C_2)$, for all pairs of clusters C_1, C_2 from the current clusters, and picking the pair whose union has minimum $CH(C)$. What we are doing is determining the variance of the newly

formed cluster and picking that cluster which will have least variance.

5 HCC for Comic Series and Instances

We apply the algorithm described above to our problem of creating simultaneously a hierarchical clustering of comic series and instances. In our case, the sets of objects we have are S, the set of comic series and D, the set of comic instances from all series. The only detail of the *HCC* algorithm specific to our problem is the relationship matrix X. We used two methods to create this matrix. They are described below.

5.1 Relationship Matrix using Bag of Words

We use a *bag of words* approach here. We run a *stemmer* [6] on the transcripts to get the stems of the words present in them. From this list of words(stems), remove *stop words*, i.e., those words which occur in almost all documents and do not convey information that we need about the comic.

5.1.1 Word and document relationship

Now, let W represent the set of all words $w_1, w_2, \ldots, w_n$ from all the documents. We create a single corpus out of all the documents in our dataset and calculate the *tfidf* of a word w_i with respect to a document d_j, based on this corpus. Lets call that value as $t(w_i)$. This value represents the relationship between d_j and w_i and hence we store it as the value of x_{ij} in the relationship matrix between words and documents.
The *tfidf* score of a word w with respect to a document d is calculated based on the corpus C from which this document was taken and is defined as follows:

$$
\begin{aligned}
D(w) &= \text{number of documents in the corpus having word } w \\
freq(w) &= \text{frequency of word } w \text{ in document } d \\
tfidf(w) &= freq(w) * \log(\frac{|D|}{|D(w)|})
\end{aligned}
$$

We run *HCC* on the words from all the documents and the documents themselves, and use the clusters of words and documents thus generated to calculate the relation between documents and series. But, when we had used this technique in our experiments, we had the words and documents as the leaves of the cluster, instead of tuples. (as mentioned previously in the description of the algorithm)

Note that this clustering will take $|W| + |D| - 1$ iterations, creating one cluster C_i in iteration i, by merging two clusters C_{i1} and C_{i2}.

5.1.2 Document and series relationship

We need to find out the relationship between documents and series. What connects them is the words in the document and the words in the series. Hence, from the clusters generated above for words and documents, we can determine the relationship between documents and series as follows.

- Initialise all values in the relationship matrix X that stores the relation between documents and series to 0.
- Let $K = |W| + |D| - 1$, where W=set of words and D=set of documents
- For clusters C_{i1} and C_{i2} that were merged in iteration i of the algorithm that was run previously for co-clustering words and documents,
  - $K = K - 1$
  - For each document d_i in C_{i1}
    - For each series s_j that each documents in C_{i2} belong to, do the following
      $$x_{ij} = x_{ij} + K$$
    - Do the same for C_{i2}

What we have done above is utilised the clusters of words and documents created by co-clustering, to obtain values for relationship between series and documents. And since the similarity decreases as we move up along the hierarchy, we keep decreasing the magnitude that we keep adding to the relationship strength between the documents and series present in any word-document cluster.

5.2 Relationship Matrix using LMs

In this case, the rows of the relationship matrix X are the documents and the columns are the series S. We use *language models* in this method. Language models are a probabilistic way of describing a corpus of text. An *ngram* is a sequence of n words in a text. Viewing all documents from a series s_i as a corpus, we create a *ngram* language model(LM) nLM_i from that corpus, where the n in nLM_i denotes how many *ngrams* were used for creating that model. We do this for all series in our dataset, thus creating as many LMs as the number of series.

The perplexity of a given text t with respect to a language model L is defined as $\sqrt[N]{\frac{1}{P(t)}}$ where $P(t)$ is the probability of t with respect to the LM L and N is the number of words in t. Perplexity is a measure of how likely this text could have come from that LM. A high perplexity means that the LM was surprised by that text. For each document d_i, we calculate the perplexity of the document with respect to the LM nLM_j created using the series s_j. Let p_{ij} denote this value. Since perplexity denotes the inverse of the likelihood of the text with respect to a LM, we decided that $\frac{1}{perplexity}$ is a measure of how related a document is to a series. Thus, we store $\frac{1}{p_{ij}}$ as the value of x_{ij} in the relationship matrix X.
We used NLTK [7] for constructing these language models.

6 Results and Discussion

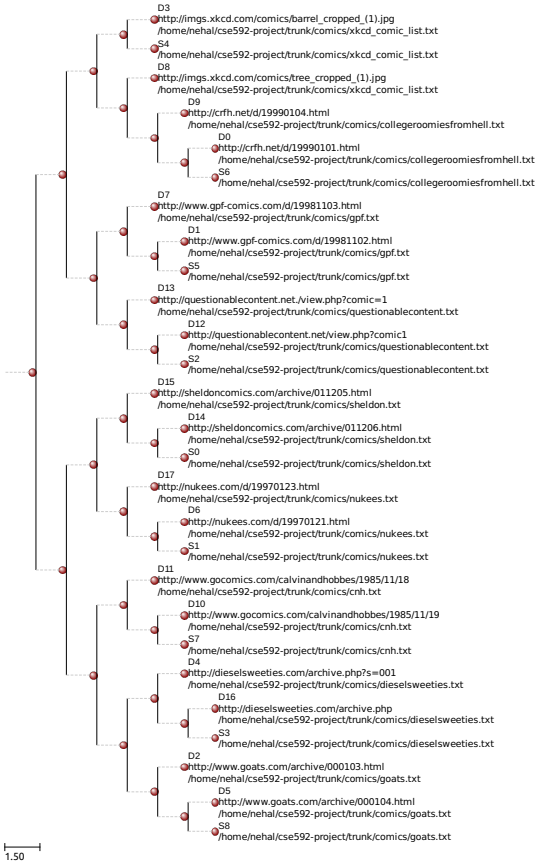

Figure 1: Dendrogram of hierarchical co-clustering using Bag of Words

Figure 2: Dendrogram of hierarchical co-clustering using 1-gram Language Model

We have shown fragments of two dendrograms. The first one is from our Bag Of Words model which used the co-clustering information from word-docuemnt cluster to create document-series cluster. The results look promising. Our technique clustered most of the comics from same series together. This indicates that comics from the same series use similar writing style, words etc. We also noticed comics from different series were clustered together. For example, instances from series $S6$ - *College Roomies From Hell* were clustered with that series, and then grouped with $S4$ - *XKCD*. This also reflects reality because they are in fact related to college life. We thus see a simultaneous clustering of comic series and instances.

The second figure shows two fragments of dendrogram generated by our second approach which used language models for each series. The first fragment shows comics from different series getting clustered together. These comics have some

similarity. *General Protection Fault* and *XKCD* are related as they have mentions of software and other related information technology terms. Both *College Roomies From Hell* and *XKCD* are related to college life. Second fragment shows the comics of same series (XKCD) grouped together. We ran this algorithm on 900 documents from 9 series.

In general, our results were satisfactory. We observed similar comics grouping together. We believe the suggestions of similar comics from this cluster would be sensisble and interesting. The success of our approach relies on the usage of similar words in different comic instances. Such similarity is quite strong in the comics from same series. However, it is not very strong across series. Still, the similarity of words is good enough to make interesting suggestions to readers. One additional advantage of hierarchial clustering is it allows users to browse the tree to look for interesting comics. Enabling users to be able to browse comics is a powerful feature which would help them to discover new content on the web.

7 Future Work

One area to explore is to use more sophisticated natural language processing techniques to obtain relationship between comic series and instances. Another improvement is that once these clusters are generated, if we wanted to add in new data, the entire process has to be re-done. That turns out to be inefficient, because new data can be added frequently, as new comics are published almost every week. To solve this, we could use the current distribution of series and instances, as prior information and use sequential updating, to determine into which cluster, new instances and series should be added to.

References

[1] Jingxuan Li et al, *HCC: A Hierarchical Co-clustering Algorithm*

[2] T. Eckes et al, *An error variance approach to two-mode hierarchical clustering*

[3] OhNoRobot `http://OhNoRobot.com`

[4] XKCD comics `http://xkcd.com`

[5] GoComics `http://gocomics.com/calvinandhobbes`

[6] Porter Stemmer `http://tartarus.org/martin/PorterStemmer`

[7] NLTK http://nltk.org