MOVSOM-II - analyzis and visualization of movieplot clusters

Stefan Gabrielsson and Sam Gabrielsson
{stga8437,saga7635}@student.uu.se

Uppsala University, Department of Information Technology (2004)

Abstract

In the paper [1] we presented MOVSOM , a fully automatic movie recommender system with no need of user preferences whatsoever. In contrast to movie recommender systems like MovieLens[1], that are depended on user input in form of rankings of movies or IMDB's[2] recommending system that relies heavily on users recommendations and keywords in order to be able to recommend movies, MOVSOM doesn't require any input. Our aim was to cluster movieplots and visualize the results in a user friendly way and make it possible for a user to select a movie they have seen and then be recommended movies that are similar to that movie in some way. Our belief that this was possible was due to WEBSOM [2], which is a system for clustering documents based on their content. In this paper we present MOVSOM-II, a successor to MOVSOM, but with better visualisation, preprocessing techniques and analyzing of the result.

INTRODUCTION

Previous work on clustering movies without user preferences can be found in [1], [3]. Despite the fact that they both fail in some sense, they both suggest that to get better result, one should use movieplots with more text and preprocessing techniques that are more sophisticated. To achieve this we have implemented some extra features in MOVSOM [1]. The new features we have implemented are the use of Porters stemming algorithm, better stoplist (more domain specific) and the use of TF-IDF as word frequency weights. The data sets are also different, we use both Usenet movie reviews (because they contain more text) and IMDb's movieplots together with taglines and title. We have also extended the possibility to analyze the result we get, this includes labelling the nodes in the SOM grid in different ways. The labeling also helps to visualize the result in a more user friendly way

THE VECTOR SPACE MODEL

We have chosen to use the vector-space model to represent the data since it is a very common way to represent such data and it is very simple to implement. In the vector-space model each movieplot is represented as a vector in a vector space. Each dimension of the vector corresponds to one keyword and the value represents the frequency of the keyword. When the data is represented in this way it is easy to calculate the similarity between the vectors and if you know which vectors are similar to each other, you can order those vectors together in clusters.

FEATURE VECTOR MATRIX

To represent the frequency of the keywords we have used the Term Frequency - Inverse Document Frequency (TF-IDF), which is a better way than the frequency formula used in [1]. The difference is that the previous formula was too sensitive to word spamming and didn't weigh in both the importance of the keyword in both the movieplot and the movieplot collection, it only considered how important the keyword was in the movieplot collection.

The formula for calculating the weight for a keyword is:

$$w_{ij} = TF_{ij} \cdot IDF_i$$

where the measure of how important the keyword k_i is in document *j* is

$$TF_{ij} = \frac{f_{ij}}{\max_z f_{zj}}$$

f_{ij} = number of times keyword k_i appears in document d_j

and the measure of how important the keyword is in the document collection is

$$IDF_i = \lg\left(\frac{N}{n_i}\right)$$

N = number of documents in whole collection

n_i = number of documents containing keyword k_i

[1] http://movielens.umn.edu/

[2] http://www.imdb.com/

Suppose we have a finite set of keywords $K = \{k_1, k_2, ..., k_M\}$ and a finite set of documents $D = \{d_1, d_2, ..., d_N\}$ then the document frequency matrix will look as follows where w_{ij} is the keywords TF-IDF weight:

$$\begin{array}{cccc} & k_1 & \dots & k_N \\ d_1 & w_{11} & \cdots & w_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ d_M & w_{M1} & \cdots & w_{MN} \end{array}$$

NORMALIZATION

To avoid domination of some dimension when we calculate the similarity of the vectors, their values must be normalized. There are many ways of normalizing a feature vectors matrix, we chose to normalize the variance [4] as it is a simple normalization method that is well used. To normalize the variance the standard deviation and mean value for each column is calculated and each value in the feature vectors matrix normalized using the formula:

$$x' = \frac{x - \bar{x}}{\sigma_x}$$

where $\bar{x}$ is the mean and σ_x the standard deviation for the column that value x is in.

DIMENSION REDUCTION

Term frequency matrices are typically of very high dimensionality (of orders of several thousands) and are very sparse (a very small percentage of the attributes of the feature vectors are non-zero), and this leads to certain problems for clustering algorithms. The problems is that it is computationally heavy and time consuming to work with feature vectors of high dimensions. Another problem is that clustering algorithms that uses Euclidian distances measures wont do so well as in high dimensions distances don't vary very much. Typically one solves this problem with the help of dimension reduction techniques. The one we have chosen is called random projection and is described in [5], [6], [7]. The main advantage with the use of random projection is that it is much faster than other well known and used techniques such as Principal Component Analysis and Single Value Decomposition. Empirical tests [5], [6], [7] have shown that random projection does well in comparison with the PCA and SVD techniques.

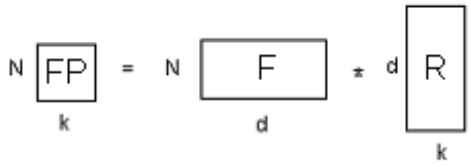

fig. 1 illustrates the process of Random Projection.

Consider a Feature Vector matrix $F_{N \times d}$ (N being number of movieplots, and d number of keywords extracted in our case) and a Random Matrix $R_{d \times k}$ (where k is the desired new dimension of the feature vectors matrix). If k is at least of size 100,according to [5], then the relation between the projected movieplots in the resulting matrix $FP_{N \times k}$ are a very good approximation of the corresponding relation to the original movieplots in the original matrix $F_{N \times d}$. Each row in the Random Matrix $R_{d \times k}$ consists of randomly generated values, empirical tests in [5], [6] have showed that it is enough to use 5 uniformly distributed 1's on each row and zero's everywhere else, to get a good approximation.

The projection is thus a simple matrix multiplication: $FP_{N \times k} = F_{N \times d}$ x $R_{d \times k}$

SIMILARITY MEASURE

To be able to determine if two vectors are similar in some sense, we need to define a similarity function. Two very common similarity functions are the Euclidian distance and the cosine similarity function.

Euclidian distance

The similarity between two documents is their distance to each other, if the distance is near zero they are considered similar and if the distance is close to one they are considered different.

The distance between two documents is calculated with the following formula:

$$\text{dist}(d_i, d_j) = \sqrt{\sum_{h=1}^{k} (d_{ih} - d_{jh})^2}$$

Cosine similarity
The similarity between two documents is the cosine of the angle between them (see fig 2), if the cosine of the angle is close to one they are considered similar and if the cosine of the angle is close to zero they are considered different.
The cosine between two documents is calculated with the following formula:

$$\text{sim}(d_i, d_j) = \frac{\sum_{h=1}^{k}\left(d_{ih} \cdot d_{jh}\right)}{\sqrt{\left(\sum_{h=1}^{k} d_{ih}^2 \cdot \sum_{h=1}^{k} d_{jh}^2\right)}}$$

and since $\sum_{h=1}^{k} d_{ih} \cdot d_{jh}$ is the same as the standard vector dot product $\vec{d}_i \cdot \vec{d}_j$ and $\sqrt{\sum_{h=1}^{k}\left(d_{ih}^2\right)}$ is the same as the norm $\|d_i\|$, we get the following formula: $\text{sim}(\vec{d}_i, \vec{d}_j) = \frac{\vec{d}_i \cdot \vec{d}_j}{\|d_i\| \cdot \|d_j\|}$

and that is the same as: $\cos\theta = \frac{\vec{d}_i \cdot \vec{d}_j}{\|d_i\| \cdot \|d_j\|}$

which is the cosine formula.

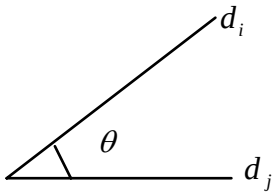

fig 2. The cosine similarity between two document feature vectors is the cosine of the angle between them.

TEXT PREPROCESSING

Before we can construct our vector-space model, the data has to be prepared for it, and this is called preprocessing of text and this is where the most work is done. This is done in several different steps and the purpose of it is to clean the text from nonsense and unimportant words and to reduce the dimension of the document vector since it tends to get very big and very sparse. As an example, table 1 show the number of word left after each preprocessing step has been performed on 95 IMDb plots.

Text preprocessing step	#Keywords*	#Words**
Text cleansing	2612	8677
Removal of short words	2597	8247
Lowercasing words	2597	8247
Removing stopwords1	2395	4399
Removing stopwords2	2335	4133
Stemming	1964	4133
Word reduction	623	2595

table 1: Preprocessing 97 IMDb plots.
* Number of unique words in the document collection.
** Total number of words in the document collection.

Text cleansing
Removes punctuation characters such as \ ' " ´ ` / . , = : ; ( ) { } ! ? -] and words containing characters other than letters.

Removal of shortwords
Gets rid of short words that the text cleansing leaves, i.e. removing the apostrophe from "Guard's" leaves the short word "s" that should be removed. We do not worry about accidentally removing any significant short words as we don't know of any.

Lowercasing words
Make documents look pretty by lowercasing all words.

Removing stopwords1
A generic text stopwords list is applied that removes words such as "the", "a" etc.

Removing stopwords2
A domain specific stopwords list is applied that removes such words as "movie", "actor" and so forth that we don't believe contribute to describing a movie. This list isn't complete and could be more extensive, though we hesitate to make it bigger than necessary initially before we are more certain about its necessity.

Stemming
Porters stemming algorithm is applied to (hopefully) stem words to their base stem, e.g. "zombie" and "zombies" might be stemmed to "zombi" which is good, however "anime" and "animal" might get stemmed to "anim" which is bad.

Word reduction
Removes words that occur to infrequently or too frequently in the document collection. This helps get down the dimensions which is necessary and thus has tradeoffs and removes word that can't possibly contribute to the clustering of document. In the above example we only removed words that only occurred in one documents.

CLUSTERING ALGORITHMS

We have used two different cluster algorithms: K-means and Kohonen's self-organizing map.

K-Means

K-Means is an iterative clustering algorithm in which items are moved among set of clusters until the desired set is reached. The algorithm consists of the following steps:

1. Choose K initial cluster randomly (representing the K movieplots) and calculate the mean for each cluster.
2. Present every movieplot vector to the algorithm and calculate the similarity between the movieplot vector and the centre of each cluster. Assign the movieplot to the cluster with greatest similarity.
3. Recalculate the means for every cluster.
4. If the means are different from previous step, repeat step 2-4. Otherwise terminate the algorithm.

We used Matlab's own K-means algorithm and we used cosine as our similarity function.

The quality of K-Means clusters can be studied by creating a silhouette plot that shows how much the clusters created are "separated" from each other .

Kohonen's self-organizing map (SOM)

The trained SOM is a two-dimensional, ordered lattice (a simple grid) where each node in the lattice represents a set of feature vectors in input space. Nodes in the lattice that are neighbours are also a neighbours in the input space.

The SOM algorithm places a set of reference vectors into the input space so that the feature vectors are approximated by the reference vectors. The reference vectors are constrained to a two-dimensional regular grid that forms an "elastic network" which follows the distribution of the training data in a nonlinear fashion.

The SOM algorithm obtains simultaneously *a clustering* of the feature vectors onto the codebook vectors and a *nonlinear projection* of the feature vectors from the high-dimensional input space onto the two-dimensional, ordered lattice formed by the reference vectors and their corresponding nodes. [2]

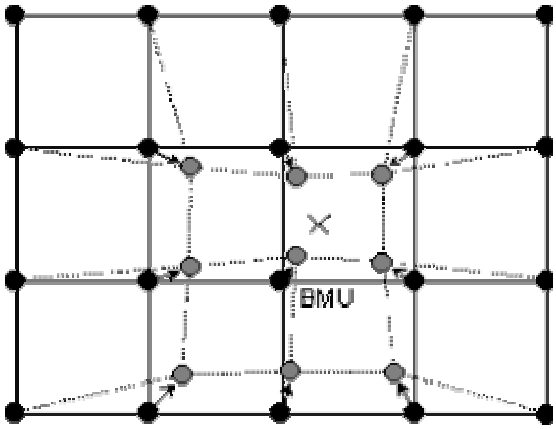

fig 3. The feature vector X is closest to the lattice node marked BMU for Best Matching Unit. The BMU itself as well as its topological neighbours are moved closer to the input vector X in the input space i.e. the input vector attracts them.

The Algorithm

The feature vectors are presented to the SOM in random order, possibly several times. At each step the best-matching codebook vector (*winner*, also called *best-matching unit* or BMU) for the current data sample is searched. That is, for each node, *j*, on the grid, compute the Euclidian distance, d_j, between its reference vector and the input vector:

$$d_j = \sum_{i=1}^{N} \left(x_i - w_{ji}\right)^2$$

where

N is the number of inputs

x_i is the value of input *i*

w_{ji} is the reference vector from input *i* to node *j*

The winning reference vector and its neighbours on the lattice are updated by the following formula:

$$w_{ji} \leftarrow w_{ji} + \eta f\left(j,k\right)\left(x_i - w_{ji}\right)$$

where

η is the learning rate

f(j,k) is the neighbourhood function

The neighbourhood function is often taken as the Gaussian function, a bell-shaped function. The width of the neighbourhood function is decreased monotonically during the learning process. Initially a large number of reference vectors are updated for each feature vector and later only a few reference vectors are slightly adjusted. In the final stage the distribution of the SOM reference vectors in the input space roughly approximates the density of the input data. One important thing to notice is that the density approximation and the ordering of the data are competing goals between which the algorithm

makes a compromise that depends e.g. on the final width of the neighbourhood function. The number of clusters need not be the same as the number of nodes - several neighbouring nodes may form a cluster. When the algorithm reaches a stable state, each reference vector express a weighted average of the data points in that map region, particularly of data points mapped to the node associated with the reference vector. The neighbourhood function defines the size and shape of the weighting function. If the neighbourhood width is zero the algorithm is equal to the K-means clustering algorithm.

We used The SOM TOOLBOX [4] in Matlab with default parameters, only varying the size of the SOM and the type of lattice used (for this report we used a rectangular lattice, but we have found the hexagonal lattice to be much better in some circumstances, but have not yet drawn any conclusions on what is best to use in our case).

The quality of a SOM can be studied in many ways, we chose to study the Unified Distance Matrix (U-Matrix) and Elastic Grid. The U-Matrix illustrates how close to each other the SOM nodes lie, due to the nature of the SOM this gives an idea of what clusters exists in the input data. The Elastic grid similarly gives an idea of how the SOM grid is stretched out in the input space (i.e. how it "looks") which also helps when trying to determine if any clusters have been found.

VISUALIZATION

To visualize the result we used the two-dimensional map that the SOM produces, K-means doesn't have that ability which is a drawback with K-means, you only get clusters with hopefully clear borders but with no information about how close the clusters are to each other or how elements inside the clusters relate to elements in other clusters.

LABELLING

A really great thing with the two-dimensional map is that you can label the nodes. We used different labels for the nodes for the same map depending on what we wanted to visualize and those where: Movie titles, common-and significant keywords, and movie genres. Labelling the nodes with keywords gives you a hint of why they where clustered together. The most interesting type of labelling is the keyword labelling in which we study the preprocessed list of keywords for each document in a SOM node and try to pick out significant keywords to use for labelling the SOM.

Significant keywords labels
The intention with this node labelling method is to show a set of keywords that are considered representative of all the documents that have the node as their BMU. This is done by creating a document collection that includes in addition to the BMU in question also all documents in directly neighbouring nodes. Feature vectors are created for the document collection using TF-IDF as weighting and only using keywords that the BMU in question contains (in order to not label the BMU with keywords it doesn't contain). The sum of the TF-IDF weights for each keyword is taken and the 5 keywords with the highest TF-IDF sum is used as labels for the node. This gives a ok labelling of the nodes that can give a good idea of what documents a node contains and how the document of one node relates to its neighbour.

Common keywords labels
This approach is very simple to implement, the keywords all documents in a node have in common are used for labelling. This approach has the drawback that sometimes all documents in a node don't have any keyword in common (very common, perhaps 3 documents reside in a node and the first two share keywords and the last two share keywords, but not the first and last document), however it usually gives a very clear indication of why - for some nodes - documents end up together.

RESULTS

We started out using 95 IMDb movie plots and 97 Usenet movie reviews as our dataset. These are very small datasets and conclusions drawn from them aren't likely to hold for large datasets. We wanted to analyze the results we get when using a small dataset to see if we get any good results, after this we would be willing to once again try large data sets. We compared the SOM clusters against IMDb's top ten recommendations for the movies "The Thing" and "Kill Bill: Vol.1". We also let two humans, using only the movie titles from the dataset containing 95 IMDb movie plots, try to form clusters in the data. For our random sample we let our three year old niece pick some movies in her own way from the list of 95 movies (she drew some very pretty pictures![3]). See *table 1* for a compilation of the recommendations the different sources gave for the movies. It is hard to draw any conclusions other than that the results vary, however our SOM clusters managed to cover the human clusters quite well.

See *fig 3-1* and *fig 3-2* for a visualization of the SOM created for the dataset consisting of 95 IMDb movie plots. The first figure shows what documents the SOM nodes contain, the second figure uses a labelling method in which significant keywords for the documents in each SOM node is extracted and used for labelling. The keyword labelling gives an idea of why the documents have ended together in a

[3] http://www.pcpinball.com/movsom/random_clustering.jpg

node (they're likely to share some of the significant keywords), and also how the documents relate to neighbouring node (which they're also likely to share significant keywords with).

We also created a SOM for 97 Usenet movie reviews. The usenet movie reviews contain much more text than the IMDb movie plots. Our intention and expectation was that we would get better clusters using larger text documents. However, we found it hard to draw any conclusions about which SOM was better. Both SOM's seemed fine but neither contained any clear clusters as in neither case the U-Matrix indicated anything but a big cluster of all documents. The map neighbourhood's of documents was quite interesting to study though. Unfortunately the documents seemed to cluster together often on very useless words "go", "take", "try". Our hope had been that those seemingly non descriptive words would together be able to capture the movie's plot, for example we might have the following two movie plots:

"**zombies** walk around not able to put two and two together"
"**vampires** walk around not able to put two and two together"

Those two plots contain two "colourful" words that describe the movies quite well, however zombies and vampires are two different words, our hope had been that the remaining rather non-descriptive words that both plots share exactly would cause them to be clustered together, unfortunately this doesn't seem to be the kind of end result we should expect, or the plots used were not "professionally" enough written such that this can happen.

We also created a SOM consisting of 545 movies assembled by picking the top 50 movies in each of IMDb's 19 genres (the genres overlap). Our intention this time was to label the SOM nodes using the movie genres, if movies in a node were of different genres we would vote for which genre to label the node with (the most common one). The genre "drama" was unfortunately three time more common among the movies and we get many drama clusters, but there seemed to be a slight grouping of movies belonging to the same genre but we found it hard to draw any conclusions about the existence of genre clusters.

We also created a SOM for 1029 IMDb plots and 1000 Usenet movie reviews. Neither SOM's U-Matrix indicated any clear clusters. The SOM's gave some few nice groupings of documents but mostly didn't give the type of recommendations we had hoped for. The documents seemed to again group on rather useless and non-descriptive words.

The only time we used Random Projection to reduce the dimensionality of feature vectors was when we worked with the Usenet dataset of 1000 movies. The document feature vectors were in this case of dimensionality 7833, when we tried to train a SOM on these feature vectors without first having reduced their dimensionality using random projection we ran out of memory (after 1 hour of computations). When we reduced the dimensionality to 314 using Random Projection we managed to train the SOM and get seemingly acceptable results. For all other datasets we were able to train the SOM without first projecting the feature vectors onto a lower dimension. We thought the SOM's we got from the non-random projected feature vectors were better, although they weren't useless in comparison.

	Recommendation for Kill Bill Vol. 1	*Recommendations for The Thing*
IMDb	Kill Bill: Vol. 2 Cidade de Deus Daredevil Freddy Vs. Jason Gangs of New York Godfather Trilogy: The Last Samurai, The Lord of the Rings: The Return of the King The Gladiator Terminator 3: Rise of the Machines	Alien³ Alien: Resurrection Dreamcatcher Se7en Predator Predator 2 Dawn of the Dead (2004) From Beyond Blade II Day of the Dead (1985)
Human1	Big Lebowski, The Face/Off Fight Club Se7en	Alien³ Alien: Resurrection Dreamcatcher Alien
Human2	Kill Bill: Vol. 2 From Dusk Till Dawn	Dreamcatcher Psycho (1960) Nightmare On Elm Street, A Exorcist, The Halloween 5 Halloween H20: 20 Years Later Freddy Vs. Jason
Random	Cidade de Deus Daredevil Freddy Vs. Jason Catch Me If You Can Day of the Dead Devil's Advocate, The Face/Off Final Destination Forrest Gump	Se7en Matrix, The Midnight Mulholland Dr. Ring, The RoboCop Rurôni Kenshin: Meiji kenkaku roman tan Total Recall
*K-Means **	Alien: Resurrection Se7en Alien³ Kill Bill: Vol. 2 Alien Aliens Matrix, The 28 Days Later...	Blade II Predator Predator 2 Daredevil American Psycho 2: All American Girl Peeping Tom Cannibal Holocaust American Werewolf in London, An
*SOM 1 ***	Predator Predator 2 Kill Bill: Vol. 2 ***close recommendations* ****** Terminator 3: Rise of the Machines Matrix, The Halloween 5 Cannibal Holocaust Ring, The Se7en Wild at heart	Dreamcatcher Alien First Blood Inferno Suspiria ***close recommendations* ****** Last Samurai, The Aliens American Psycho 2: All American Girl E tu vivrai nel terrore - L'aldilà Phenomena Gladiator Psycho Cruel Intentions Map of the World, A Videodrome
*SOM 2 ****	***close recommendations* ****** Kill Bill Vol.2 Gladiator From Dusk to Dawn Tenebre Under the Tuscan Sun	Alien3 Godfather Trilogy ***close recommendations* ****** Predator 2 Aliens Resurrection Aliens Halloween 5 From Dusk Till Dawn 3: The Hangman's Daughter Peeping Tom Donnie Darko Deranged Annie Hall

table 2: Recommendations from different sources for two different movies.
* 95 IMDb movies plots were clustered using K-Means into 10 clusters, initial cluster centroids were picked randomly among the feature vectors.
** 95 IMDb movie plots were used to create a SOM of size 8x8.
*** 97 USENET movie reviews were used to create a SOM of size 8x8.
**** The SOM is able to recommend not only movies that it considers to be very similar to a specified movies, it can also recommend movies that it thinks are quite similar, i.e. "close recommendations" (these recommendations are found in nodes that are neighbours to the node containing i.e. The Thing).

- Nightmare On Elm Street, A		- American Werewolf in London, An	- Se7en	- Predator - Predator 2 - Kill Bill: Vol. 1 - Kill Bill: Vol. 2		- From Dusk Till Dawn	- Twin Peaks: Fire Walk with Me
- Batoru rowaiaru	- From Dusk Till Dawn 3: The Hangman's Daughter		- Terminator 3: Rise of the Machines - Matrix, The - Halloween 5 - Cannibal Holocaust - Ring, The	- Wild at Heart		- Freddy Vs. Jason	
	- Dawn of the Dead - Lord of the Rings: The Return of the King, The - Blade Runner - RoboCop	- Blade II	- Face/Off - Day of the Dead	- Alien³ - Cidade de Deus - Godfather Trilogy: 1901-1980, The		- Vana espuma - Star Wars: Episode VI - Return of the Jedi - Basket Case - Funhouse, The - Forrest Gump - Superstition	
- Texas Chain Saw Massacre, The	- Caged Heat - 28 Days Later...	- Brood, The - Big Lebowski, The	- Alien: Resurrection - From Beyond - Blue Velvet - Angel - Truck Stop Women - Return of the Living Dead, The - Catch Me If You Can - Tenebre	- Daredevil - Rurôni Kenshin: Meiji kenkaku roman tan: Tsuioku hen - Total Recall - Final Destination - Halloween H20: 20 Years Later - Doors, The - Peeping Tom - Terms of Endearment	- Gangs of New York - Mulholland Dr. - People vs. Larry Flynt, The - Play It Again, Sam	- Lost Highway	- Donnie Darko - Sleepy Hollow
	- Artificial Intelligence: AI						- Annie Hall
- Black Christmas		- Under the Tuscan Sun	- Last Samurai, The - Aliens - American Psycho 2: All American Girl - E tu vivrai nel terrore - L'aldilà - Phenomena	- Dreamcatcher - Thing, The - Alien - First Blood - Inferno - Suspiria	- Gladiator - Psycho - Cruel Intentions - Map of the World, A - Videodrome	- Manhattan	- Fight Club
- Exorcist, The	- Midnight						- Twin Town
- Blood Sucking Freaks - L.A. Confidential		- Deranged	- 21 Grams		- Devil's Advocate, The - Sweet Hereafter, The		- Ice Storm, The - Anything Else

fig 3-1. SOM for 95 IMDb movie plots with document labels as node labels.

- real - girl - friend - wear - nightmar		- human - world - american - die - surviv	- hunt - world - time - take - victim	- kill - hunt - bill - see - time		- famili - take - mysteri - priest - mexico	- life - town - twin - seri - follow
- forc - friend - last - wear - island	- human - vampir - forc - surviv - isol		- world - see - time - american - life	- hunt - killer - goe - hire - mother		- take - year - decid - dream - jason	
	- human - futur - blade - forc - man	- human - world - vampir - blade - prei	- take - live - day - dead - scientist	- live - kill - world - alien - human		- life - murder - young - death - go	
- end - home - power - famili - eat	- futur - day - big - take - escap	- man - big - day - daughter - real	- take - dead - live - scientist - human	- take - live - friend - dead - scientist	- murder - young - new - take - death	- life - murder - young - death - mysteri	- life - death - go - nearbi - famili
	- home - old - real - life - end						- meet - relationship - love - romant - serv
- girl - old - dead - make - polic		- decid - home - friend - life - old	- friend - young - school - time - discov	- school - friend - life - young - discov	- friend - school - life - love - new	- love - live - meet - relationship - friend	- meet - live - go - club - idea
- girl - littl - sick - old - make	- girl - polic - home - mother - dead						- live - love - try - help - young
- girl - differ - real - polic - littl		- take - live - mother - di - unnatur	- marri - world - mother - learn - live		- famili - young - join - lose - lawyer		- citi - try - help - life - new

fig 3-2. SOM for 95 IMDb movie plots with significant keywors as node labels.

TOOLS

We wrote a number of tools in order to aid the creation of the SOM, we also used a two MATLAB toolboxes, the statistics toolbox and the SOMTOOLBOX [4].

Text preprocessing tools

The text preprocessing was implemented using a set of PHP scripts that can be called from the command line. Each script operates on a directory containing a set of text documents, the text documents all have numerical names which makes it easy to identify them. The text documents are preprocessed one by one by each script and written back to the directory they were found in (or to a separate directory, which enables analysis of how the documents change during the preprocessing). Each script performs a certain type of preprocessing, but accepts command line arguments that make them behave differently for different data sets.

regexp.php

Applies a set of regular expressions to each text file in a directory. The script takes as argument the path to a text file containing perl regular expressions, reads in these expressions and applies them to the text files in a directory. If no regular expression file is provided a default list of regular expressions is loaded that removes first punctuation and then all words containing non alpha characters (basically only words with numbers since punctuation has been removed). Regular expressions makes it easy to remove punctuation characters, words containing numbers, headers from usenet documents and so forth.

shortwords.php

Removes words of a specified length from each text file in a directory. The script takes as argument the minimum allowed length of words. Typically the minimum allowed word length is 2 and is only used to remove one letter words that are a sideffect of the removal of punctuations, i.e. "Guard's" without punctuation is "Guard", "s", so the "s" would be removed. Additionally word's such as "I" and "a" are removed, though those are stopwords and will be removed anyhow if a suitable stopwords list is applied.

lowercase.php

Lowercases all words in each text file in a directory. This is useful to perform before applying a stopwords list that doesn't consider the word case, otherwise it only make the textfiles look prettier.

stopwords.php

Removes words that are "stopwords" from each text file in a directory. The script takes as argument the path to a text file containing a list of words that are to be considered stopwords.

stemming.php

Applies porters stemming algorithm to all words in each text file in a directory.

wordlist.php

Creates a file containing statistics for all keywords in each text file in a directory. Is useful for studying how the text preprocessing reduces the amount of words in a document collection.

Feature vectors matrices tools

Feature vectors are stored in SOMPAK format, which is a plain ascii file containing information about a feature vectors matrix including all its weights and row and column labels. The SOMPAK format is compatible with the Matlab SOMTOOLBOX which has a functions for reading in files in this format. Operations on feature vectors matrices are done mostly using matlab precompiled executables which save the results in both SOMPAK format and Matlab's internal .mat format.

docfv.php

This PHP script is used to read in the final preprocessed text files in order to create a feature vectors matrix where each row corresponds to a document feature vector and each column corresponds to a keyword, the values in the matrix are the keywords frequency in each document. The document feature vector matrix is stored in SOMPAK format.

transpose.exe

A Matlab precompiled executable that takes as argument a SOMPAK file containing a feature vectors matrix, transposes it and writes the resulting matrix back to a file in SOMPAK format. The program is typically used to transpose a document feature vector matrix into a keyword feature vectors matrix where the rows correspond to a keyword and each column correspond to a document.

tfidf.exe

A Matlab precompiled executable that takes as argument a SOMPAK file containing a feature vectors matrix where the values are keyword frequencies. The matrix is converted to a matrix where the values are TF-IDF weights.

normalize.exe

A Matlab precompiled executable that takes as argument a SOMPAK file containing a feature vectors matrix and the name of the normalization method to be used. The possible normalization methods are var, range, log, logistic, histD, histC which are all described in the SOMTOOLBOX documentation for the som_normalize method.

randprj.exe

A matlab precompiled executable that takes as argument a SOMPAK file containing a feature vectors matrix and the dimension to reduce the feature vectors matrix to. The program uses the Random Projection algorithm (with 5 ones on each row) to reduce the dimensionality of the feature vectors.

Clustering tools

Matlab m-files were used to create clusters based on a SOMPAK file containing a feature vectors matrix. It was more comfortable to work within matlab and use its graph window to visualize certain parts of the clustering than to create stand alone matlab executables that can't display any graph windows.

gen_som.m

This matlab m-file contains the function gen_som() which essentially takes as argument the path of a SOMPAK file containing a feature vectors matrix and the size of the SOM to create and what type of lattice to use. The function will use the SOMTOOLBOX to create the SOM and to generate a image of the SOM's U-matrix and Elastic Grid. Additionally the function will create a text file (som-bmus.data) containing the size of the SOM and which SOM node each feature vector has as Best Matching Unit (BMU), this file is the only file needed to visualize the SOM outside matlab.

gen_kmeans.m

This matlab m-file contains a function gen_kmeans() which essentially takes as argument the path of a SOMPAK file containing a feature vectors matrix and the number of clusters the kmeans algorithm is to find and what type of initialization and distance measure it is to use. The function will use the matlab kmeans() function to cluster the data and the matlab silhouette() function to generate a Silhouette plot for the clusters. Additionally the function will create a text file (kmeans.data) that contains each feature vector and what cluster it got assigned to, this file is the only file needed to study the clustering of the data outside matlab.

Labelling tools

The labelling of the SOM nodes and also of the K-Means cluster is done using a PHP script that will operate on the files the gen_som.m and gen_kmeans.m m-files generate.

label.php (som version)

Reads in a som-bmus.data file that contains a list of document id's and their BMU, from that data the following set of labels files are created:

- *som-labels-bmucount.data:* Specifies the label of each SOM node to be its BMU count, i.e. the number of times it is the Best Matching Unit for some document.
- *som-labels-doclabels.data:* Specifies the label of each SOM node to be the name of the documents that have the node as their BMU. Each document's docid will be translated to a text string using a for the dataset specific index file that translated docid's to doclabels.
- *som-labels-commonkey.data:* Specifies the label of each SOM node to be the set of keywords all the documents that have the node as their BMU have in common.
- *som-labels-signkey.data:* Specifies the label of each SOM node to be the set of most significant keywords for the node, i.e. the set of keywords that best represent the documents that have the node as their BMU.
- *som-doclist.data:* Additionally a file containing all the documents in the SOM is created, it contains for each docid, its corresponding doclabel and bmu. This is a useful reference list when visualizing the SOM.

label.php (kmeans version)

Reads in a kmeans.data file that contains a list of document id's and the cluster id of the cluster they belong to, from that data the a similar set of labels as that for the SOM is created.

Visualization tools

The visualization of the SOM and the K-Means clusters is done using a php scripts that generate dynamic html pages. The php scripts simply read in a labels file and display its contents in a nicely formatted html document, they contain no logic whatsoever otherwise.

CONCLUSION

Despite the fact that we improved the preprocessing of text and that we used documents with more text we couldn't draw the conclusion that we did better this time. Indeed, when we used K-mean we could get some nice clusters and for the smaller sets we got maps that showed some nice neighbourhoods of movies, but we never got a indication in the U-Matrix that distinct clusters exist. We implemented a number of different labelling techniques in order to evaluate why documents end up together and found this very helpful and a good indication that it is valid to talk about neighbourhoods of documents that are similar. Perhaps these are the best results we can get without creating an extensive stopwords list that excludes all "bad" words, but as mentioned we might not want to do that. In any case a too extensive stopwords list might result in very biased clusters. An alternative to stopwords list we believe would be to only pick out words that are nouns during the preprocessing, which is something we believe that might be a interesting thing to try out.

The SOM's created in this report can be found online at the MOVSOM[4] website.

[4] http://www.pcpinball.com/movsom/

REFERENCES

[1] : S & S. Gabrielsson, "Movie clustering using the self-organizing map: MOVSOM"

[2] : K. Lagus, "Text Mining with the WEBSOM". Acta Polytechnia Scandinavia, Mathematics and Computing Series No. 110, Espoo 2000

[3]: M.Fleischman, E.Hovy, "Recommendations without user preferences: a natural language processing approach

[4]: J.Vesanto, J.Himberg,E.Alhoniemi & J.Parhankangas, "SOM Toolbox for Matlab 5", Report A57, April 2000

[5]: Kohonen, T. ,"Self-organization of very large document collections: State of the art." In Niklasson, L., Bodén, M., and Ziemke, T., editors, *Proceedings of ICANN98, the 8th International Conference on Artificial Neural Networks*, volume 1, pages 65-74. Springer, London, (1998).

[6]: T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. "Self Organization of a Massive Document Collection." *IEEE Transactions on Neural Networks*, Special Issue on Neural Networks for Data Mining and Knowledge Discovery, volume 11, number 3, pages 574-585. May 2000.

[7]: E. Bingham and M. Heikki ,"Random projection in dimensionality reduction: Application to image and text data" Laboratory of Computer and Information Science, Helsinki University of Technology, Finland.