An information theoretic approach to determining sparsity in clustering classified documents

A senior project submitted to the Division of Mathematics and Computer Science,
Bard College
by Maksim Tsikhanovich
Submitted on April 27, 2011

Contents

1 Motivation & Introduction

A fundamental requirement for being able to understand natural language seems to be the ability to differentiate between words, terms, and phrases. Since the the problem of differentiating essents has remained open since at least Aristotle, it seems that differentiating the words that are use to refer to essents is a philosophically open problem. Thus when the problem of differentiating between natural language documents is represented using standard mathematics, one of the difficult problems becomes defining a function that measures similarity between words, terms, phrases, and finally documents.

In particular, in document clustering there are two problems: creating cluster centers that are representative of the document set, and determining how similar documents are to those centers. Thus a clustering algorithm must find good cluster center candidates, and assign documents to clusters whose center is similar to the documents according to a similarity measure. The problem of defining the similarity measure depends on how documents are represented, yet the restriction to some representation does not seem to make the problem of selecting a similarity measure easier.

Research papers on clustering algorithms (Lee and Seung 2001), (Blei et al. 2003), (Ng, Jordan, and Weiss 2001) typically have the following structure:

1. present or assume a document representation and corresponding similarity measure between documents,

2. provide a objective function that should be optimized in order to achieve a good clustering,

3. present an algorithm that is guaranteed to converge to a local minimum/maximum of the objective function,

4. provide a measure of the quality of the algorithm.

Since clustering is a hard problem, each one of the above parts is also difficult. However while parts 1-3 are clearly tightly interdependent, part 4 is slightly less dependent on the others. Yet being able to measure the quality of a clustering algorithm would is very important for doing 1-3, and it is why I am interested in this part of the research.

In the literature there are at least three types of quality measures:

1. *Internal measures* that depend only on the clusters as the algorithm has produced and sees (for examples of typical measures see (Zhao, Karypis, and Fayyad 2005)); typically algorithms are designed to optimize some internal measure, so using internal measures to compare algorithms that optimizes different internal measures seems to not make sense. Similarly rather than comparing algorithms that use the same internal measure as their objective function against one another, it is usually better to combine those algorithms to create a hybrid stronger than any of the individual algorithms. (Ding, Li, and Peng 2006)

2. *External measures* that compare the clusters output by the algorithm to the hand annotated classes in which documents lie. A popular measure is purity, which is the average percentage of documents from only one class in each cluster. An other well known measure is normalized mutual information, which assess when one is given the cluster from which a document originates, how much information do they gain about the class it was from.

3. *Relative measures* rely on measuring differences between at least two clustering solutions. Rand uses his c similarity measure (Rand 1971) to evaluate how well clustering algorithms detect natural clusters, or respond to perturbations in data points. Alternatively Rosell et. al. (Rosell, Kann, and Litton 2004) take advantage of having two versions of human classifications of the same data set. They evaluate the difference between the two human classifications, and say that a clustering algorithm's output is valid if it is within that distance from one of the human algorithms.

The problem that I am exploring is when given a collection of documents, wherein the documents are split among classes, how can we summarize what the documents of each class tell us about the class. Document clustering solves this problem, because the cluster centroids should be representative of the documents within their clusters. Thus the goal is to create good clusters, and therefore the problem is to determine what a good cluster is.

The data set I am working is 11675 product reviews about twelve types of products from the online electronics store Newegg. Each product review is annotated with the type of product that it is about, the review rating (whether other shoppers found that product review helpful), and its date. In order to improve the quality of the data, each of the 11675 product reviews used had at least a +3 rating, meaning that at least three people found it helpful. Since reviews are annotated

with the type of product type, and since I'm interested in summarizing the content for each of the product types, which in this case are the classes, the appropriate type of evaluation measures are external measures. Figure 1 displays the classes in the Newegg data set, and what proportion of it they make up.

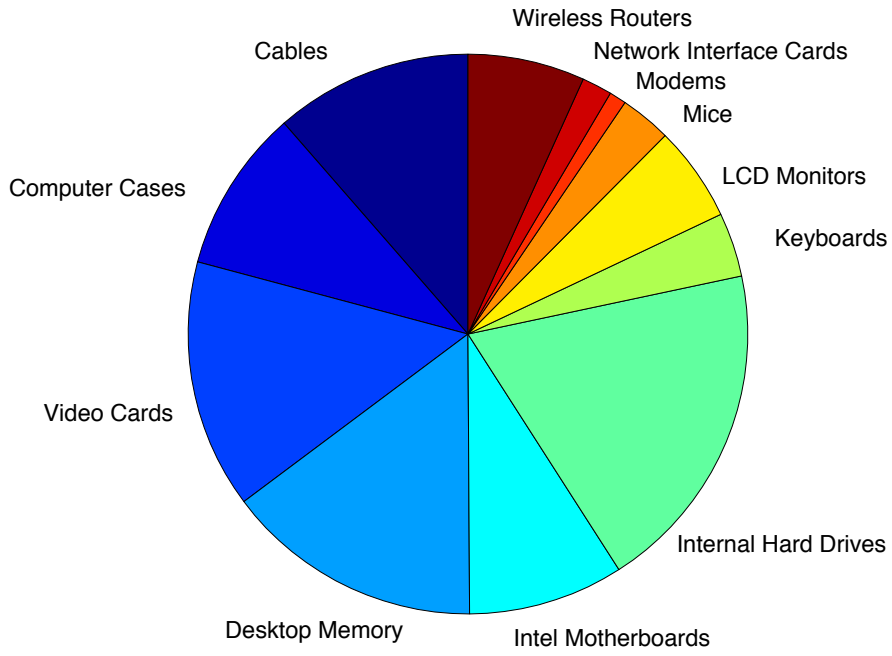

Figure 1: The 12 class Newegg product review data set, shown are the product types that the reviews are about.

The rest of the thesis is structured as follows, in Section 2 the clustering problem is defined and sparse non-negative matrix factorization is introduced as a solution. In Section 3 the evaluation problem is described, and normalized mutual information is introduced as a good solution. In Section 4 a technique is presented on how normalized mutual information is used to determine clustering sparsity, as well as number of clusters. Finally in Section 5 results from working with the Newegg reviews is presented.

2 Document Clustering

Document clustering is a open problem, and the solution that can be used largely depends on how documents are represented.

2.1 Document Representation

Usually documents are assumed to be complex statements made up of some form of atoms. The simplest atoms could possibly be letters, though intuitively the letters used in a document are not good for distinguishing it from other documents. A more complex atom is the n-gram, sets of n consecutive letters, which seem to good representatives of documents as in (Damashek 1995). The next and perhaps most intuitive atoms to consider are words. Since written language is heavily emphasized in the English tradition, most words carry a lot of meaning that is difficult to further atomize. Words are the atoms that will be used in this paper. A level above words can be structures that take advantage of sentence grammar in order to extract more meaning, such as subject verb object triples as in (Church and Hanks 1990). Once the atoms are determined they must be put together in order to reconstruct the document. The approach used in this paper is called bag of words, because in the representation of the document by words, it is assumed the document has no structure, and all we care about is the amount of times each word has been used in the document. The opposite to this would be the conceptual structures for example as defined by John Sowa, where the goal is to preserve the relationships between all words in the document.

In particular I use a stemmed term frequency-inverse document frequency (TFIDF) representation. Stemming reduces words to their linguistic stem, so for example the words give, gave, giving would all be reduced to the stem give. The effect of this is to reduce the dictionary size, or the total number of words by ignoring the differences among similar words. Furthermore since the documents I'm working with are unmoderated reviews from Newegg, there is a lot of noise, so words that occur less than five times in the document collection are ignored, because they are likely typos or otherwise unintelligible. Term frequency is intuitive, the number of the times each word occurs per document is counted; inverse document frequency is a way of weighting words: for example knowing that a document has the word 'the,' is less informative than knowing that it has the word 'coaxial' simply because 'the' is a common word, and is not good for distinguishing documents.

Before continuing it is necessary to establish some notation. The set of words used for TFIDF is called the dictionary; suppose the dictionary has d words. Then the TFIDF representation of a document will be a vector in $\mathbf{R}^d_+$, the set of vectors in $\mathbf{R}^d$ with non-negative components: $\{(v_1, v_2, ..., v_d) \in \mathbf{R}^d : v_j \geq 0\}$. Now suppose the collection contains n documents, and let $\mathrm{TF}(\mathbf{x}, v)$ denote the number of times word v occurs in document $\mathbf{x}$. Then the TFIDF value for each word v of a document $\mathbf{x}$ will be

$$\mathrm{TF}(\mathbf{x}, v) \log \frac{n}{\sum_{i=1}^{n} \mathrm{TF}(\mathbf{x}_i, v)}.$$

The term in the denominator simply counts how many times the term occurs in the collection. Thus the collection of n documents can be thought of as a non-negative matrix $X \in \mathbf{R}^{n \times d}$. Given this representation of the collection, suppose it can be factorized such that $X = WH$ for $W \in \mathbf{R}^{n \times k}$ and $H \in \mathbf{R}^{k \times d}$. Then thinking of W as the matrix which represents how documents are distributed over the k clusters, and H as the matrix where cluster centers are defined is how non-negative matrix factorization can solve the same clustering problem that soft k-means does.

2.2 Basic document clustering: k-means

The clustering algorithm that is easiest to understand is called $k - means$, creates and refines k cluster centers and assigns documents to cluster centers so as to optimize some objective function. Simple choices for the objective function are distance of each document to the cluster centroid, or cosine similarity of documents to clusters or their centroids. The basic algorithm is as follows:

1. Initialize k cluster centers. This can be done by selecting k documents to serve as centers, or simply vectors randomly selected from $\mathbf{R}^d$.

2. Greedily assign each document to a cluster center such that its assignment optimizes the objective function. If parts of a document can be assigned to more than one cluster, then the algorithm is called *soft* or sometimes *fuzzy* k-means.

3. Update the k cluster centers to be the centroids of their clusters.

4. Repeat steps 2,3 until only a certain fraction of documents are reassigned.

The three main components of k-means and most clustering algorithms are the cluster center initialization, the refinement of the cluster centers, and the choice as well as optimization of the

objective function. To see different approaches to each of the above areas a good place to start is (Zhao, Karypis, and Fayyad 2005).

2.3 Non-negative Matrix Factorization as a clustering technique

The idea that when a document collection is represented in a matrix, that factorizing that matrix can be used for evaluating similarity between documents and thus clustering them is not new, and is at least as old as Latent Semantic Analysis (LSA) (Deerwester et al. 1990). LSA computes the singular value decomposition of X as $X \approx U\Sigma V^T$ such that $U^TU = V^TV = I$ and Σ is a diagonal matrix that contains the singular values of X. LSA then proceeds to compute the rank k approximation of X, X_k by setting all but k of the largest singular values in Σ to 0. LSA was then refined with probabilistic LSA (Hofmann 1999) (or LSI, I for indexing when used in a information retrieval context), who noted that the fundamental problem with LSA was that it did not have a probabilistic foundation, and the factorization $U\Sigma V^T$ could not necessarily be used to reconstruct the original collection, since X_k could contain negative values. Probabilistic LSA operates under the assumption that documents were created by probabilistically choosing topics, and then again probabilistically choosing words from each topic to form documents. The parameters for the probability distributions are estimated using Expectation Maximization (EM) techniques, i.e. the goal is to find parameters for the topic and word distributions that maximize the expectation that the given data set was generated by the probability distribution with those parameters, for more detail on EM a good reference is (Bishop 2007).

Probabilistic LSA sounds similar to non-negative matrix factorization (NMF) in the sense that the W matrix in NMF defines a probability distribution of documents over topics if we enforce the constraint that the one norm of each row of W is equal to 1, where the *one norm* of a vector $\mathbf{w}$ is defined as $||\mathbf{w}||_1 = \sum_i w_i$. Similar constraints can be enforced on H such that its rows define the probability distribution of choosing words given a topic. In fact it can be shown that if NMF is designed to optimize a particular information theoretic objective function, then it is equivalent to pLSA (Ding, Li, and Peng 2006). In this paper I do not use this objective function though, rather I use the Frobenius error, where the *Frobenius norm* of a matrix X is defined as $||X||_F = \sqrt{\sum_i \sum_j X_{i,j}^2}$.

2.4 Rank-one Residual Iterations as a solution for Sparse NMF

In this paper the clustering algorithm used is NMF with sparsity constrains computed through the relatively new method of rank one residual iterations (RRI) as introduced by Ngoc-Diep Ho (Ho 2008). The key idea behind RRI is that when the reconstruction error $||X - WH||_F^2$ is minimized the reconstruction error contributed by each of W's columns is also minimized. The reverse is not necessarily true, nevertheless the assumption behind RRI is that minimizing the error contributed by each column of W will likely minimize the full reconstruction error.

The RRI algorithm used here is the one proposed by Ho for minimizing the reconstruction error with weighted sparsity

$$||X - WH||_F^2 + \lambda||W||_1. \tag{1}$$

Equation 1 is called the objective function. A *sparse* matrix X is one whose *zero norm* $||X||_0$, defined as the number of non-zero entries in X, is small compared to the size of X. Notice that Equation 1 does not use the zero norm because despite being called a norm, the zero norm does not satisfy the axioms of being a norm, and it is difficult to work with; thus below RRI is refined such that minimizing the one norm version of Equation 1 W will also minimize the zero norm version.

The intuition for enforcing sparsity is that it prevents overfitting, and hence NMF should create a better model of the process that created the collection we have at hand. This is similar to the idea in polynomial curve fitting, that when a high degree polynomial is fitted to data that was generated by a polynomial of lower degree the coefficients of the high degree polynomial will grow in norm, and thus enforcing a penalty on the norm of the coefficients reduces overfitting (Bishop 2007). Also enforcing sparsity helps the summary of documents in terms of topics be easier to interpret, consider Figure 2.

2.5 The RRI Algorithm

The RRI Algorithm is composed of an initialization step and refinement steps. To make the initialization more understandable, I present the refinement steps first.

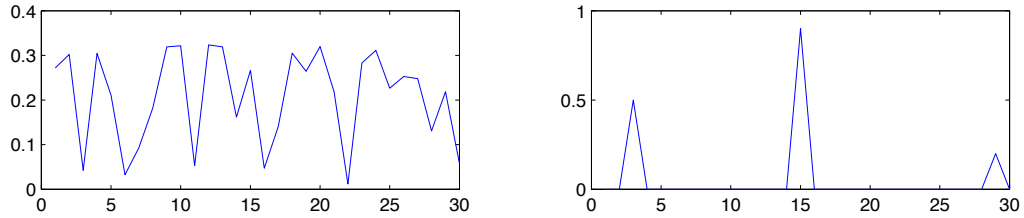

Figure 2: Suppose the above are representations of two vectors of 30 components. The one on the left is dense because it is made up of small parts of multiple components, while the one on the right is sparse because most of its components are zero while some have large magnitude (differences in magnitude due to assuming the vector represents a probability distribution). If the two vectors represent how a document is made up of topics, then the left one is harder to understand because it says that document is made up of small parts of lots of topics, while the right one says the document is largely composed of a few topics.

2.5.1 Refinement

At each iteration the column of W being optimized, w_t is updated

$$w_t \leftarrow \frac{[R_t h_t - \lambda 1_{n\times 1}]_+}{||h_t||_2^2} \tag{2}$$

where $R_t = X - \sum_{i\neq t} w_i h_i^T$ is the residual that w_t ought to explain, h_t is the corresponding tth column of H, and $[v]_+$ is the projection of v onto the positive orthant. As a result in Equations 1 and 2 λ serves a parameter which zeros out increasingly larger entries of w_t as its value increases. Note that setting $\lambda = max_i \sum_j X_{ij}$ will force all entries of W to be set to 0. After optimizing each column of W, each column of H is optimized by a update rule similar to Equation 2 and then the process is repeated again until some convergence criterion is met.

As mentioned earlier, Ho's base algorithm is somewhat incomplete due to the use of the one norm as opposed to the zero norm in Equation 1. The objective function we are interested in minimizing is

$$||X - WH||_F^2 + \lambda ||W||_0 \tag{3}$$

while Ho's algorithm attempts to minimize the function in Equation 1. The main problem this poses is scaling, namely $WH = \alpha W \frac{1}{\alpha} H$. Due to this problem RRI may decrease $||W||_1$ without actually decreasing $||W||_0$. To eliminate this possibility the rows of H are constrained to have $||h||_1 = z$, where z is a chosen parameters. This is done by solving for the optimal h_t using Ho's algorithm and then projecting onto the $||h||_1 = z$ simplex using the method of (Duchi, Singer, and

Chandra 2008). Varying z lets one trade off between reconstruction error and sparsity of H as seen in Figure 3.

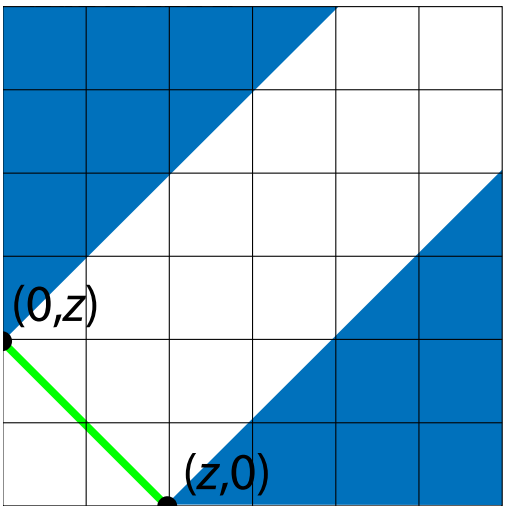

Figure 3: In 2 dimensions we are projecting points from the xy plane onto the 2-d simplex. A point in the blue region will be projected onto $(0, z)$ or $(0, z)$ and thus be sparsified. As z is made larger the amount of points that get projected onto the facet as opposed to the vertices increase and thus reconstruction error improves at the cost of sparsity.

However for the purposes of this paper varying z is not a good way to trade off between reconstruction error and sparsity in H or W. First, since we wish each row of H to define a probability distribution of topics over words, then clearly $z = 1$ is the appropriate choice, and the one that is actually used for this paper. Second it is clear that by first finding the h_t that minimizes the objective of Equation 1 and then projecting it onto the $||h_t||_1 = z$ no longer optimizes that objective. To see this consider the following example: suppose

$$f(x, y) = 1000x^2 + y^2,$$

which is a function from the same family of quadratic functions as our objective function, and $z = 1$. It is minimized at $(x, y) = (0, 0)$, so this point will be projected onto $(0.5, 0.5)$; clearly $(x, y) = (0, 1)$ is a better solution; we will examine an example in our context after we develop the optimal solution. In practice using this approximation is not this bad because the objective function for h_t is much more balanced across its independent variables (i.e. they have similar coefficient magnitudes as opposed to 1000 compared to 1), however we propose using Lagrange multipliers for correctly minimizing the objective for h_t under the constraint that $||h_t||_1 = z$.

For this purpose let us define (as Ho does in (Ho 2008)) $x = R_t^T w_t$, and then for some permutation Π, if $\Pi x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ where $x_1 \geq 0$ and $x_2 < 0$ then $\Pi h = \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$. Then we can define the objective function for h_t

$$f_{h_t}(h_t) = ||R_t||_F^2 - 2h_1^T x_1 + ||w_t||_2^2 h_1^t h_1 \tag{4}$$

The Lagrangian is

$$\Lambda(h_t, \gamma) = f_{h_t}(h_1) + \gamma(||h_1||_1 - z) \tag{5}$$

Note that we are constraining $||h_1||_1 = z$ because we will set $h_2 = 0$ (to see why refer to Ho's thesis). Now we can take $\nabla\Lambda = \left(\frac{\partial\Lambda}{\partial h_1}, \frac{\partial\Lambda}{\partial\gamma}\right)$. Setting $\nabla\Lambda = 0$ gives rise to the linear system

$$\begin{pmatrix} \alpha & 0 & \dots & 0 & 1 \\ 0 & \alpha & \dots & 0 & 1 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & \alpha & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{1|h_1|} \\ \gamma \end{pmatrix} = \begin{pmatrix} 2x_{11} \\ 2x_{12} \\ \vdots \\ 2x_{1|x_1|} \\ z \end{pmatrix} \tag{6}$$

where $\alpha = ||w_t||_2^2$ and h_{1i} and x_{1i} are the ith components of h_1 and x_1 respectively. This system can be solved quickly using linear programming methods. The problem is that this solution is generally not going to be non-negative. The non-negative solution requires using Lagrange multipliers with inequality constraints, which then require 'activation coefficients', the problem is then solved for all possible combinations of activation coefficients. In essence there are $2^{\dim h}$ solutions here that need to be considered unless there is a clever technique to determine which coefficients should be active, i.e. whether it is possible to determine based on an x which values of the optimal h would be set to zero. We are not sure whether this is possible, and therefore we use the simplex projection approximation to this problem with hopes that the worst case does not occur often.

2.5.2 Initialization

Here the problem is how to initialize the matrices W and H. RRI is very sensitive to the initialization of W and H. This is for three main reasons. First, if they are initialized in a region of $\mathbb{R}^{n\times k} \times \mathbb{R}^{k\times d}$ where the objective function is very flat without actually being at a minimum, the

algorithm may terminate too soon. Second, the NMF problem is non-convex, as can be seen by the fact that if (W, H) is a global minimum of the objective function, then so is $(QW, Q^{-1}H)$, where Q is any invertible matrix that does not change the 1-norms of the rows of H (e.g. a permutation matrix). Therefore, we expect the solution found to be sensitive to the starting conditions. Third, there is no reason to think that all local minima are global minima, and even if the first problem does not occur, the algorithm will find only local minima.

We resolve this problem by randomly selecting 4 initializations, running RRI on each, and choosing the run with the lowest objective value. We have no good reason for doing 4 initializations rather than 5, and one possible avenue of further research might be to see how many initializations one must do to have a given probability that the best run is within some distance of the actual minimum (or of the minimum over all possible runs).

Random initialization of the matrices is performed in 5 steps.

1. Assign each entry of H to be zero (with probability $1 - p$) or non-zero (with probability p) .
2. Choose the value of each non-zero entry of H from the uniform distribution on $[0, 1]$.
3. Choose the value of each entry of W from the uniform distribution $[0, 1]$.
4. Find the real number α which minimizes $||X - \alpha WH||_F$, and scale W and H by $W = \sqrt{\alpha}W$, $H = \sqrt{\alpha}H$.
5. Perform simplex projection on the rows of H.

RRI is guaranteed to converge, in the sense of having less than a certain percentage change in objective value of the solution over iterations, where one iteration is an optimization of all columns of W and all rows of H. However RRI it is not guaranteed to converge to a global minimum since the problem is non-convex. In practice, it may fail to get close even to a local minimum. A matlab implementation of the above RRI algorithm is provided in Appendix B.

3 The Cluster Evaluation Problem

Like document clustering, the evaluation of clusters is an open problem. For this paper I am restricting evaluation methods to external evaluations. Consider an external evaluation function, in general this function is given a prior collection of probability distributions of documents over classes and a computed collection distribution (the clustering). It must then compare how similar these collections of distributions are. In this paper both the prior and computed distributions are discrete, so the family of evaluation functions are functions of the form $f\colon \mathbf{R}^{nq+nk} \to \mathbf{R}$ where n is the number of documents in our collection, and q, k are the number of classes and clusters, respectively. For any n, q, k this is a large family of functions, and no one function stands as the correct choice a priori. Fortunately if we accept the axioms of information theory, then we can take advantage of the many results in the field that are useful for comparing probability distributions.

3.1 Normalized Mutual Information

The foundations for modern information theory were laid out by Claude Shannon in his 1948 paper "A Mathematical Theory on Communication." (Shannon 1948) Shannon was interested in how much information is received when someone observes a random variable X takes on value x_0 with probability $P(X = x_0)$ (I will use $P(x_0)$ to stand for this for sake of clarity); let this quantity be denoted $I(x_0)$. Shannon started with two axioms for what the information function $I(x_0)$ should satisfy:

1. $I(x_0)$ should only depend on $P(x_0)$, and as $P(x_0) \to 0$ then $I(x_0) \to \infty$ monotonically,
2. I should be additive, i.e. for two independent events x, y it holds that $I(x+y) = I(x) + I(y)$.

In his paper Shannon proves that the only function that satisfies these axioms is

$$I(x_0) = -\log_b(P(x_0)) \tag{7}$$

for some b. The *entropy* of a random variable X, denoted $H(X)$ is defined as the expected information X, for a discrete random variable with s possible states this is

$$H(X) = -\sum_{i=1}^{s} P(s_i) \log_b(P(s_i)). \tag{8}$$

Since we are interested in comparing probability distributions, we are interested in the entropy of a joint random variable, denoted as $H(X, Y)$, but first we consider the conditional entropy of random event Y given X as

$$H(Y|X) = -\sum_{i,j} P(x_i, y_j) \log_b(P(y_j|x_i)) \tag{9}$$

Then using product and sum rules for probabilities it follows that

$$H(X, Y) = H(X) + H(Y|X). \tag{10}$$

This quantity can be seen to mean that the amount of information need to encode the states that X, Y take is equal to the amount need to encode X plus the amount needed to encode each state of Y when the state of X is known.

The mutual information of two random variables X, Y as defined in (Bishop 2007) is

$$\begin{aligned} \mathrm{MI}(X, Y) &= H(X) - H(X|Y) = H(X) - H(Y) - H(X, Y) \\ &= H(Y) - H(Y|X) = H(Y) - H(X) - H(X, Y). \end{aligned} \tag{11}$$

Intuitively the mutual information of two variables X, Y measures the number of units of information (typically bits are used) that one knows about the value of Y when he or she knows about the value of X. For reasons that will become clear later, the mutual information is normalized when measuring clustering quality as in (Manning, Raghavan, and Schtze 2008) such that

$$\mathrm{NMI}(X, Y) = \frac{2MI(X, Y)}{H(X) + H(Y)}. \tag{12}$$

Note that if X, Y are independent then $H(Y|X) = H(Y)$ and

$$\mathrm{NMI}(X, Y) = \frac{2(H(X) - H(X|Y))}{H(X) + H(Y)} = 0,$$

while $H(X|X) = 0$ and

$$\mathrm{NMI}(X, X) = \frac{2(H(X) - H(X|X))}{H(X) + H(X)} = 1.$$

3.2 Computing NMI in Document Clustering

In document clustering we are interested in comparing how well the distribution of documents over classes compares to the distribution over clusters. Let C be a random variable that takes on the identity of each class with probability equal to the percentage of documents that are in that class; similarly let K be a random variable that takes on the identity of each cluster with probability equal to the number of documents in that cluster. While C is obtained from the prior annotations of the documents in the collection, the distributions in K correspond to the columns of the matrix W as defined in Section 2.1.

To define NMI, we first define $H(C)$ and where q represents the number of classes as

$$H(C) = \sum_{i=1}^{q} \frac{-n_i}{n} \log_b \frac{n_i}{n} \tag{13}$$

and where n_i is the number of documents in the ith class or cluster. Then for C, K if there are k clusters

$$H(C|K) = -\sum_{i=1}^{q} \sum_{r=1}^{k} \frac{n_i^r}{n} \log_b \frac{n_i^r}{n^r} \tag{14}$$

Where n_i^r refers to the number of documents in the ith class and rth cluster, and n^r refers to the number of documents in the rth cluster. By substituting Equations 13 and 14 into Equation 12, a formula for NMI is achieved. In (Manning, Raghavan, and Schtze 2008) NMI is defined as

$$\mathrm{NMI}(C, K) = \frac{2 \sum_{r=1}^{k} \sum_{i=1}^{q} \frac{n_r^i}{n} \log_b \frac{n_r^i}{n_r n^i}}{\sum_{i=1}^{q} \frac{-n^i}{n} \log_b \frac{-n^i}{n} + \sum_{r=1}^{k} \frac{-n_r}{n} \log_b \frac{-n_r}{n}}. \tag{15}$$

3.3 NMI as a Clustering Evaluation Measure

Since the NMI measure comes from a very large family of evaluation measures, a important question to ask is why is this measure used. This section describes how NMI is useful when evaluating clustering that is used for document summarization. In particular the NMI measure poses three desirable qualities:

1. NMI is founded on information theory, thus the NMI measure of a clustering solution has meaning,

2. NMI is responsive to changes in the sparsity parameter λ,

3. NMI is responsive to changes in the number of clusters k.

While the first point becomes significant when the summary created by the clustering algorithm is used for extrapolating characteristics of the document collection, the focus of this paper is on the second and third points.

3.3.1 NMI and Sparsity

First we discuss the second point, that NMI is responsive to changes in the sparsity parameter λ. Recall that RRI NMF works by updating all k columns first of the matrix W using the update rule stated in Equation 2, and then it uses a similar update for the k columns of H^T. This process is supposed to minimize the objective function (Equation 1) which is composed of the Frobenius reconstruction error and weighted sparsity of the W matrix. After this process is completed the NMI of the solution W and the original classes can be measured; in this section we show that NMI as affected by changes in λ.

We begin with an example. Suppose we are using RRI NMF to factor a matrix $X \in \mathbf{R}^{5\times d}$ as $X = W_0 H$, and suppose the following is the W_0 we receive for a particular λ_0:

$$W_0 = \begin{pmatrix} \boxed{0.43626} & 0.05223 & 0.084398 & 0.075976 & 0.35113 \\ 0.34673 & 0.10661 & \boxed{0.37154} & 0.16145 & 0.01367 \\ 0.037394 & 0.16104 & \boxed{0.28186} & 0.26966 & 0.25005 \\ 0.22373 & \boxed{0.23454} & 0.11889 & 0.19405 & 0.22878 \\ 0.15669 & \boxed{0.23908} & 0.1983 & 0.23775 & 0.16818 \end{pmatrix} \text{Classes} = \begin{pmatrix} 1 \\ 3 \\ 3 \\ 2 \\ 2 \end{pmatrix} \tag{16}$$

This is a generally good solution, because looking at the mode (the boxed values) of each document's distribution we can see that modes are grouped together by class. However it is a largely confusing solution, because for some documents the modes are not much higher than the other values, and as a result $\text{NMI}(W_0, \text{Classes}) = 0.0623$, a relatively low value. Now suppose we set $\lambda_1 = \lambda_0 + \Delta\lambda$, for $\Delta\lambda = 0.05$. Recall the update function in RRI NMF (Equation 2) is

$$w_t \leftarrow \frac{[R_t h_t - \lambda 1_{n\times 1}]_+}{||h_t||_2^2}$$

where w_t and h_t are the tth columns of W and H^T, respectively. In our case we have

$$w_t \leftarrow \frac{[R_t h_t - (\lambda_0 + \Delta\lambda)1_{n\times 1}]_+}{||h_t||_2^2},$$

and in this example this means that all values of W_0 are decreased by $\Delta\lambda$, and if any value becomes negative, it is set to 0. Then the rows of W_0 are normalized to have one norms equal to 1, and we call this new solution W_1. In our example

$$W_1 = \begin{pmatrix} \boxed{0.43626} & 0.05223 & 0.084398 & 0.075976 & 0.35113 \\ 0.35154 & 0.10809 & \boxed{0.37669} & 0.16369 & 0 \\ 0 & 0.1673 & \boxed{0.29281} & 0.28013 & 0.25976 \\ 0.22373 & \boxed{0.23454} & 0.11889 & 0.19405 & 0.22878 \\ 0.15669 & \boxed{0.23908} & 0.1983 & 0.23775 & 0.16818 \end{pmatrix} \tag{17}$$

The key thing to notice is that the modes of the rows that had values which were zeroed increased in magnitude, while other rows remained completely unchanged. As a result, this solution is slightly more correct, and in fact we have $\text{NMI}(W_1, \text{classes}) = 0.0649$.

As a second example, we examine how λ affects NMI for clustering on a well behaved dataset. For this we use a subset of the TDT2 dataset, which is a collection of news articles on approximately 100 different topics. The subset used is the one that contains the 30 largest topics by number of documents on that topic. Figure 4 summarizes the result.

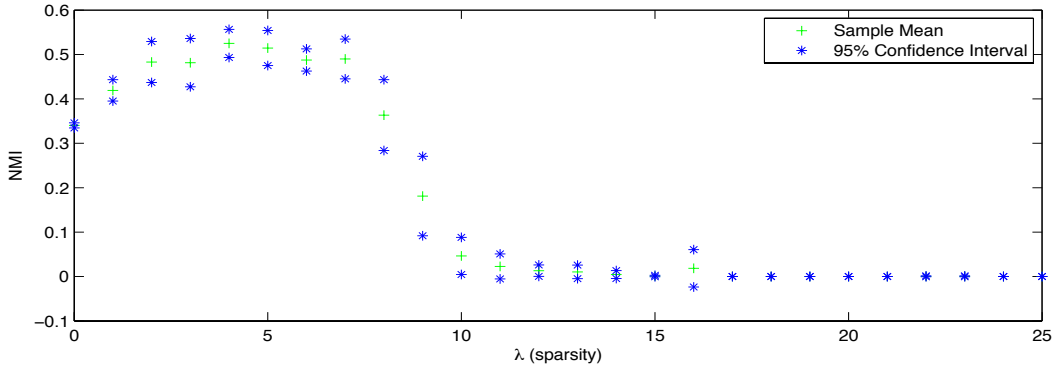

Figure 4: TDT2. For this clustering we use $k = 30$, because we know apriori that there are 30 classes, so 30 clusters should be reasonable. For each value of λ displayed 15 clusterings were computed, displayed are the mean NMI values of those clusterings as well as one standard deviation. Notice how the highest NMI values are achieved on the $\lambda \in [3, 7]$ interval.

Finally for the dataset of Newegg reviews, NMI exhibits a similar trend as λ varies, but the trend is much more messy because the product reviews are written in many different writing styles, and unique grammars. This trend can be seen in Figure 5. Notice that the range of the λ parameter is much larger for this dataset than for the TDT2 dataset. This imposes the constraint on our solution that it shouldn't be sensitive to scale, i.e. a solution which increments lambda by a fixed value until some optimal value is achieved would not be acceptable.

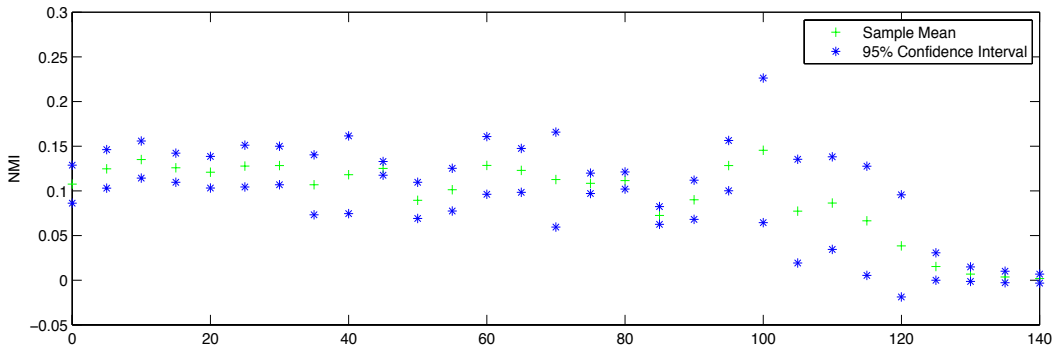

Figure 5: Newegg twelve classes. For this clustering we use $k = 13$, because there are 12 classes, but as will be explained in Section 3.3.2, 13 may be better. For each value of λ displayed 15 clusterings were computed, displayed are the mean NMI values of those clusterings as well as one standard deviation. Notice how the highest NMI values are achieved on the $\lambda \in [180, 280]$ interval.

3.3.2 NMI and Number of Clusters

Intuitively, the number of clusters that will maximize NMI should be the number of classes from which documents occur, however often NMI isn't maximized for this exact number, thus we still want to select the optimal number of clusters using NMI. This is not because the number of classes is unknown, since then we are not able to evaluate NMI as we defined it.

To intuitively understand why a number of clusters different from the number of classes may yield a higher NMI score than equal numbers, we examine the Newegg dataset. There seem to be many reviews that inherently difficult to classify as being about a particular product, so they will be difficult to group with reviews that are easy to classify. The following three reviews are examples of the difficult to classify ones:

- Works great. No problems. None.

- Good value. Fast. Runs at advertised speeds with no problems or errors. Cant think of one thing.
- Got the item quick, has been working great since I installed it. No problems Nothing.

While the following reviews are relatively easy to classify and would thus be put into clusters with similar reviews:

- Large Capacity, Low noise, reasonably Fast none so far, 3 months in heavy use for desktop. (Internal Hard Drives)
- This memory was packaged in something that seemed bulletproof, lets you know its not going break during shipping. Looks nice, deep blue color [...] (Memory)
- This allows for perfect transition from DVI out to HDMI in. The cord is very long and durable. (Cables)

The reason that it may be better to have a number of clusters that differs from the number of classes is that some of the clusters can be used to collect the reviews that are hard to classify. To illustrate this, suppose we had the above six reviews and a clustering solution that put each of the easy to classify reviews in separate clusters while the ones that were difficult to classify were evenly split among the existing three clusters, this would look something like

$$W_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0.33333 & 0.33333 & 0.33333 \\ 0.33333 & 0.33333 & 0.33333 \\ 0 & 1 & 0 \\ 0.33333 & 0.33333 & 0.33333 \\ 0 & 0 & 1 \end{pmatrix} \text{Classes} = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \end{pmatrix}. \tag{18}$$

In this case we have $\text{NMI}(W_0, \text{Classes}) = 0.2103$. Now suppose we added an extra cluster, and now all documents that were spread evenly among the initial three clusters are split evenly among all

four clusters, as in:

$$W_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0 & 1 & 0 & 0 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{19}$$

in this case $\text{NMI}(W_1, \text{Classes}) = 0.2171$, a small improvement over the previous result. In general as k increases RRI-NMF has the property that documents that are similar will remain in clusters together while documents that were different but formerly in the same cluster will be split up, as a result most clusters will become more pure, and thus have a better NMI. In the above example if the difficult documents were somehow recognized as being similar, and placed in one cluster, the increase in NMI would have been dramatic.

The normalization in NMI is there to penalizing having too many clusters. Since as $k \to n$, each document will have its own cluster, so a simpler cluster purity measure would be maximized by setting $k = n$. Due to the way that NMI is defined, dividing by half the sum of the entropy of the classes and the clusters will enforce NMI's range to be the interval $[0, 1]$. Figure 6 displays NMI's response to changes in k.

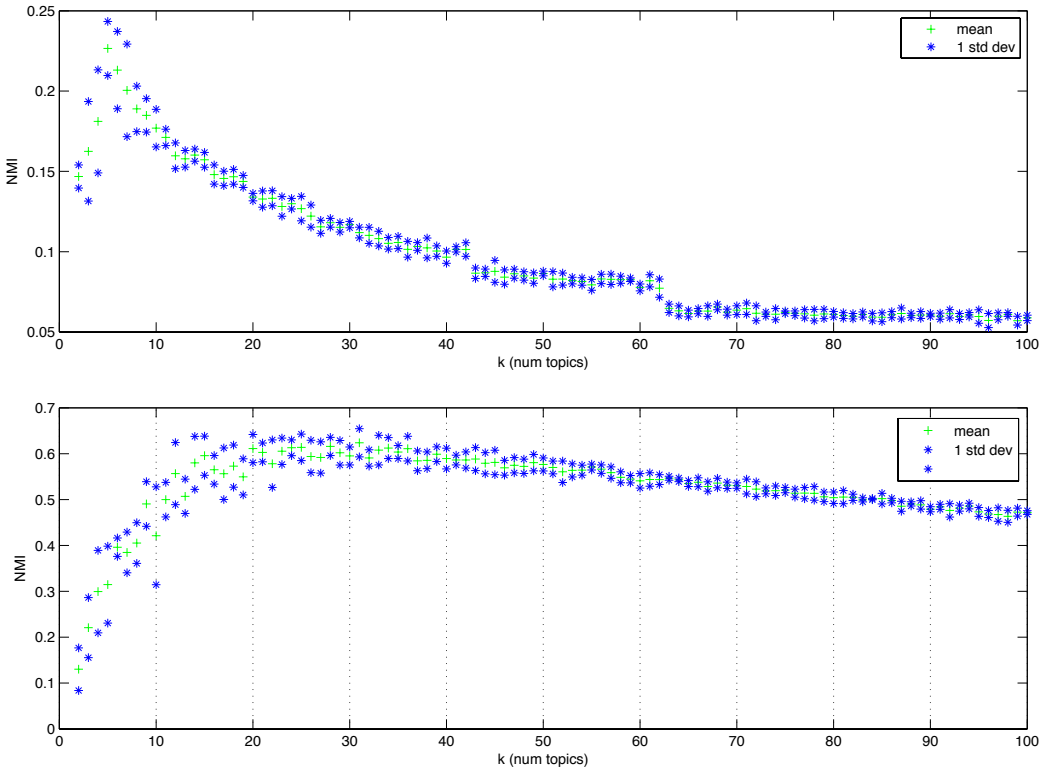

Figure 6: The Newegg, and TDT datasets, respectively. Notice that as k increases, so does NMI initially, however after a certain value NMI begins to decrease. Our goal is to find this critical value.

4 Using NMI to Select Optimal Sparsity and Number of Clusters

Since NMI measures how well a clustering solution represents the classes that documents originally were from, we wish to select sparsity and number of clusters that will optimize the NMI measure, and thus provide the best clustering solution. The rest of this section is divided into two subsections. In the first subsection we show than NMI has a cubic dependence on λ. In the second subsection we take advantage of this dependence to develop a fast and accurate method for determining the λ that maximizes the NMI for a particular distribution.

4.1 Dependence of NMI on λ

It is possible to try and compute the partial derivative of NMI with respect to λ. However this process is not very illuminating, and is difficult; for more see the Appendix. In this section we use the more expedient approach of fitting polynomials to data in order to see that NMI has a cubic dependence on λ.

In order to fit a polynomial we need to determine what the interval should be, and how we should sample the interval. From Equation 2 we know that $\lambda_{\max} = \max_n ||X(:,n)||$ and $\lambda_{\min} = 0$. To first see the shape of how NMI responds to change in λ we sample points equidistant from one another along the interval. We have already seen what the result looks like in Figure 5, and this shape leads to the guess that the depends of NMI upon λ can be represented well by a cubic function. The cubic fit to the Newegg data, as well as the TDT2 data is displayed in Figure 7.

4.2 Determining Sparsity and Number of Clusters

Once the cubic approximation of the NMI-λ function is available, the extreme value theorem can be applied to infer that the optimal NMI occurs either at $\lambda = 0$ or at one of the critical points of the cubic. A original constraint was that the solution's complexity should not be a function of the sizes of the input collections, and hence of λ. In order to achieve this the number of distinct λ at which the NMI is sampled needs to be low, and the particular λ need to be chosen well. A well known result in numerical analysis says that to optimally sample $n+1$ points that will fit to a nth

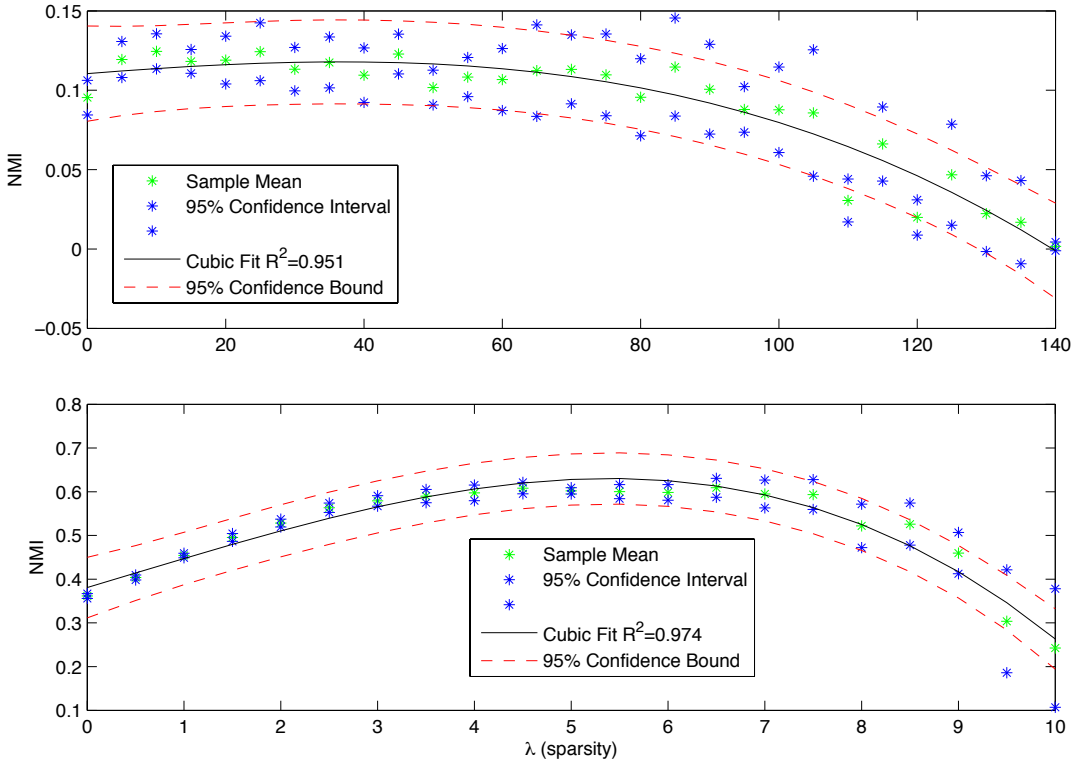

Figure 7: The Newegg, and TDT datasets, respectively. Notice that most of the observations about the NMI value at each λ fall into the 95% confidence bound to the cubic function, and that the cubic functions have good R^2 values.

degree polynomial we should have

$$\lambda_i = \frac{\lambda_{\max}}{2}\left[1 + \cos\left(\frac{2i+1}{2n+2}\pi\right)\right], \; i = 0, \ldots, n. \tag{20}$$

In order to determine the number of clusters, we follow the simple principle that given a number of classes that documents in the collection originate from, this number is close to the number of clusters that will maximize NMI. Thus a algorithm to find the optimal k and λ is the following:

1. Let k be in $\{k_0 - m, k_0 - m + 1, \ldots, k_0, \ldots, k_0 + m\}$.

2. For each k compute the optimal λ for that k, let this be λ_k.

3. For each k and λ_k calculate the NMI n times, and suppose the 95% confidence interval around the maximal NMI achieved for (k, λ_k) has width $2\epsilon_k$.

4. For k where the maximum NMI of k and λ_k is highest among all (k, λ_k) pairs keep taking samples of it until the overlap between this solutions confidence intervals and the next best solution's confidence interval is less than some threshold. If the k changes for which the maximum NMI is highest, repeat this step for that k.

5. Finally select (k, λ_k) such that the 95% confidence interval around its maximal value overlaps less than a threshold parameter with the second best (k, λ_k) candidate.

In theory we expect this algorithm to converge because the true standard deviation of the NMI for different k and λ_k should be the same, so as more samples are taken then changes in ϵ_k will eventually be the same. In practice a limit is placed on the number of times step 4 is iterated, and a result that should be close to the true result is used. Refer to Appendix B for a matlab implementation of this function.

5 Experiments with NMI's Effect on Clustering

In this section we examine what clustering solutions with different NMI values look like. In order to do this we will look at clustering solutions for both the Newegg dataset, and the TDT2 dataset. For the TDT2 dataset we examine a solution with the highest NMI achieved by our algorithm, and a solution whose NMI we know can be better. For the latter two the only difference in the solutions will be in the λ and k parameters.

5.1 TDT Dataset

In Figure 8 we start by looking at the solution space that the λ, k optimizing algorithm sees for the TDT dataset. We compare the clustering results of a solution with NMI=0.39, and a solution with

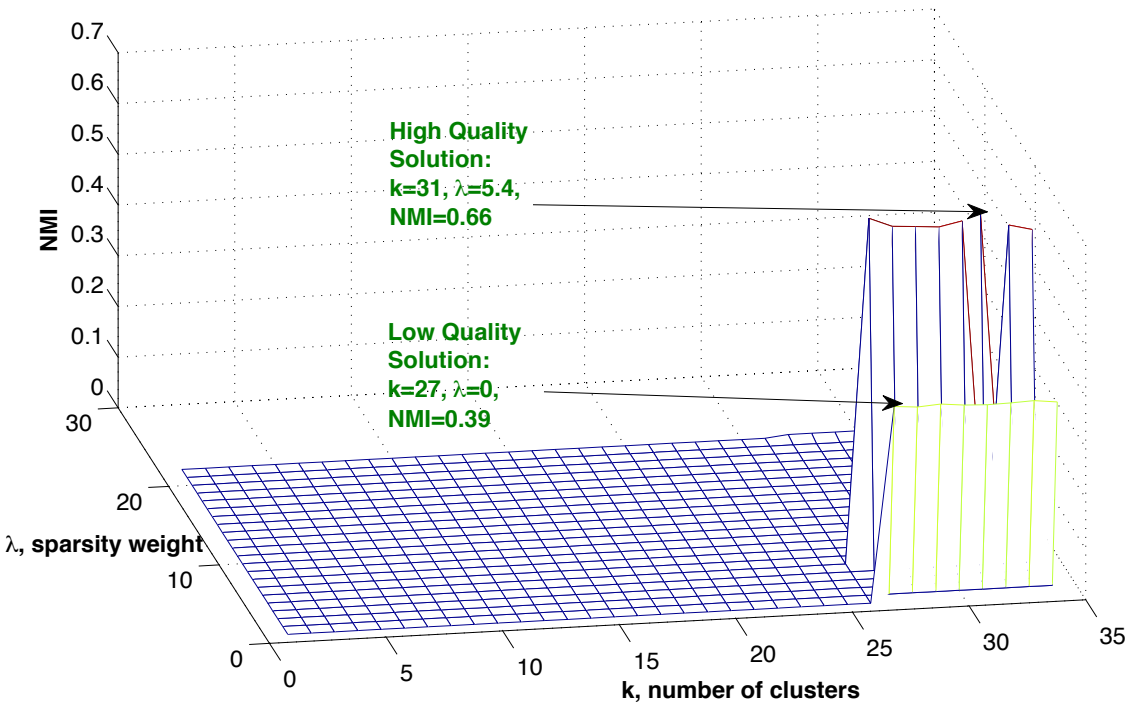

Figure 8: Note that even though $\lambda \approx 20$ was explored for all values of $k \in \{27, 28, \ldots, 34\}$, the NMI at those solutions was equal to zero. The rest of the NMI=0 area is simply unexplored.

NMI=0.62. The clusterings are presented in Figure 9.Note that these representations are screen-shots of a flash application, the full applications are accessible at `http://turing.bard.edu/~mt166/SP/`. The most interesting thing about these two representations is that they look very similar, the main

difference being how individual clusters evolve over time, i.e. the heights of the bands that represent clusters vary at different rates. This is largely an effect of how this representation responds to changes in λ. The shown representation presents the most significant words and documents for each cluster, while changes in λ affect words and documents with low significance, and increases in lambda tend to simply make the significant words and documents more significant. A simple measure of the quality of a clustering is called Purity, which measures to what extent each cluster is composed of only documents from one class, and for a clustering Ω and class distribution $\mathbb{C}$ it is defined as

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k} \max_{j} |\omega_k \cap c_j|. = \frac{1}{N} \sum_{i=1}^{q} \max_{r} n_r^i \tag{21}$$

For the solution with NMI=0.62 the purity is 0.62, while the purity of the NMI=0.39 solution is 0.42. We expect there to be a correlation between NMI and Purity because both have the n_r^i factor. However as will become apparent when examining the Newegg dataset, the max function that is part of purity complicates this relationship slightly.

5.2 Newegg Dataset

Once again to begin, we consider a picture of the solution space that the λ, k optimizing algorithm sees in Figure 10. Figure 11 displays what the clustering solutions look like for the low quality, high quality on the Newegg dataset. The first thing that stands out is that neither the NMI=0.08, nor the NMI=0.23 solutions seem to be good. Both solutions use words such as '1080p' or 'drive' to define multiple topics, whereas to a human expert it is clear that if the clustering should reflect the original class structure, which has individual classes for monitors and hard drives, then these words should be used to define unique clusters. In a good solution clusters would be uniquely defined in terms of keywords, and it should be clear what the cluster to class relationship is.

What is peculiar about these clustering solutions is that the solution with lower NMI has higher purity. In particular the NMI=0.08 solution has purity=0.38 while the NMI=0.23 solution has a purity=0.33. Generally this is possible because even though both purity and NMI contain factors of n_r^i, the NMI is dependent on other factors as well, so that even when a solution's purity is lowered, some of the other factors of NMI may be increased by a larger amount.

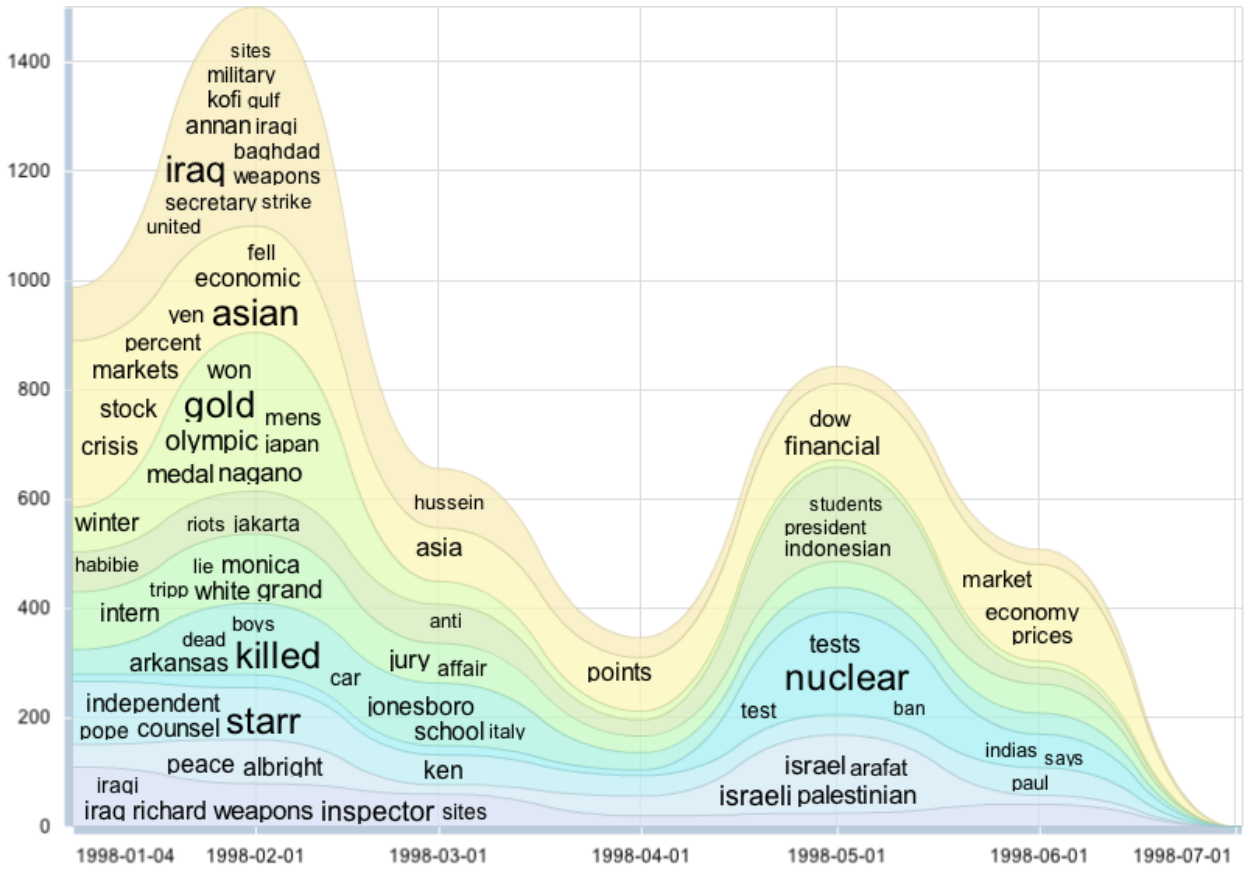

Figure 9: TDT2 dataset clustering solution with NMI=0.39, $\lambda = 0$, $k = 27$. The bands of different color represent clusters, and the height of each band at a particular date represents how many documents from that date were determined as belonging to that particular cluster. The words that appear in the bands are the keywords that describe clusters, and the magnitude of the words varies with how important they were for describing documents from that cluster at a certain time.

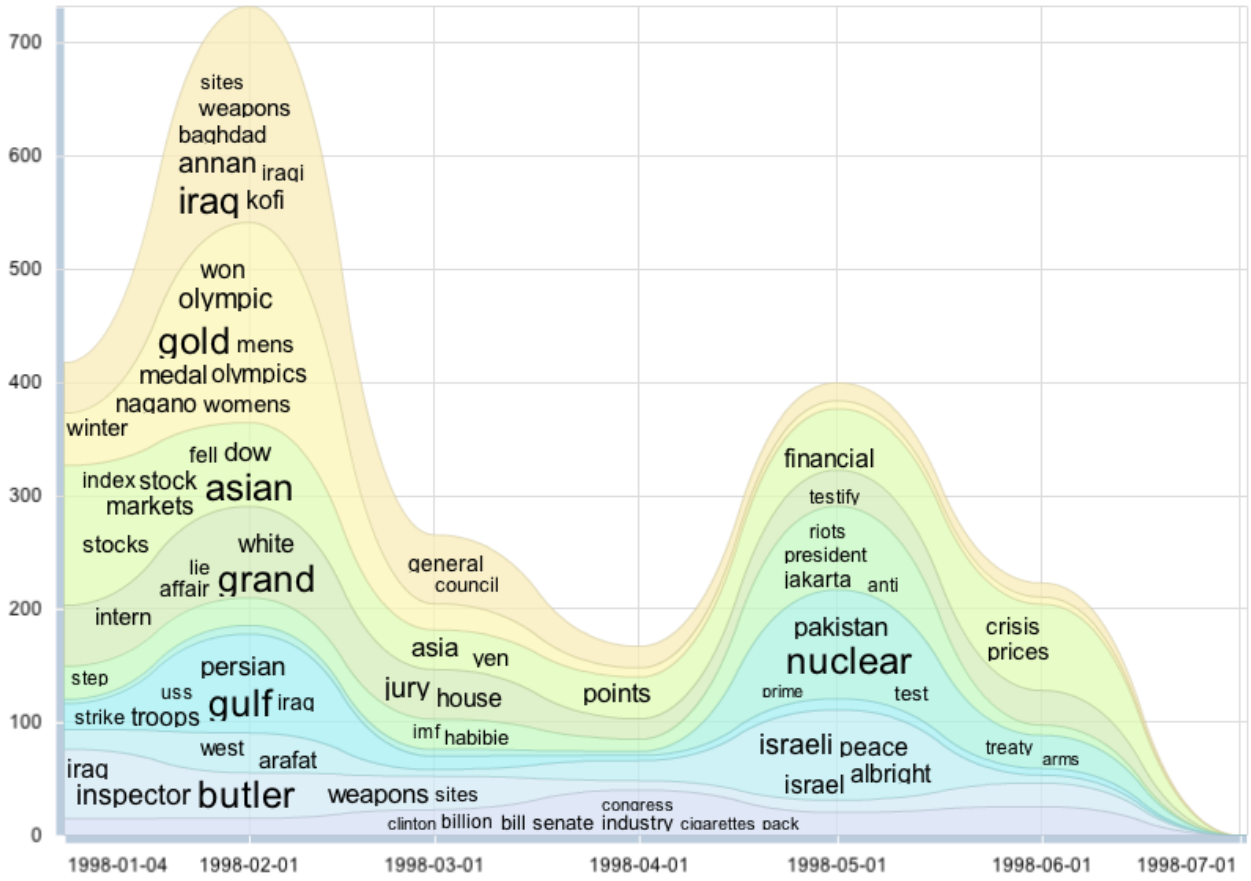

Figure 9: TDT2 dataset clustering solution with NMI=0.62, $\lambda = 5.4$, $k = 31$.

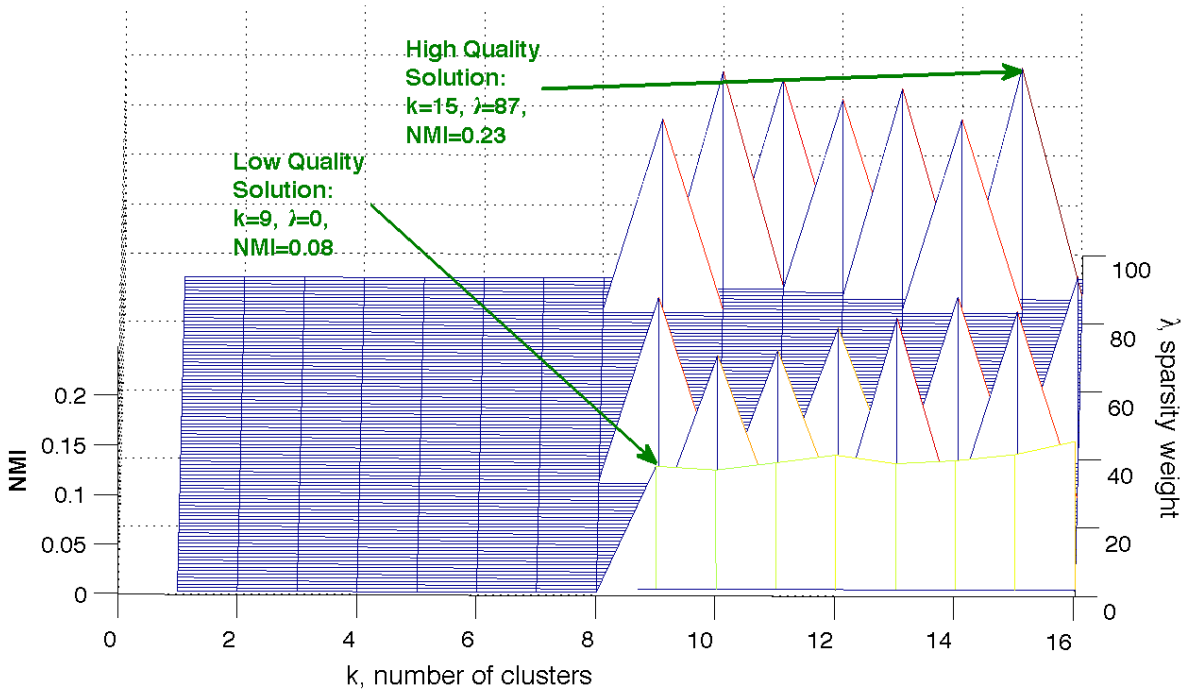

Figure 10: Note that for different values of k, slightly different values of λ are explored, and that change in λ has a large effect on change in NMI than does change in k. The second effect is simply due to the fact that the complete range of λ values is displayed, while only a small range of k values is displayed, and this is because k values for away from the number of classes do not yield solutions with high NMI.

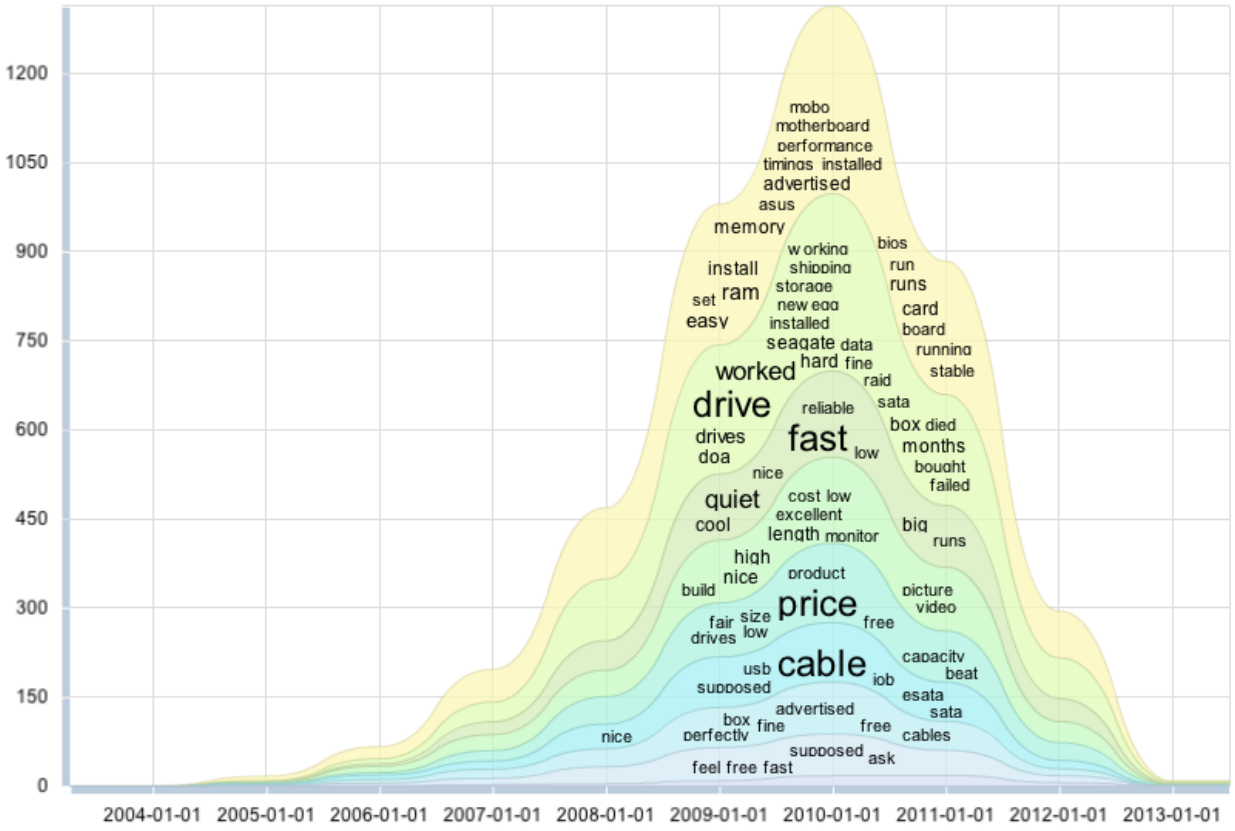

Figure 11: Newegg 12 reviews dataset with NMI=0.08, $\lambda = 0$, $k = 9$.

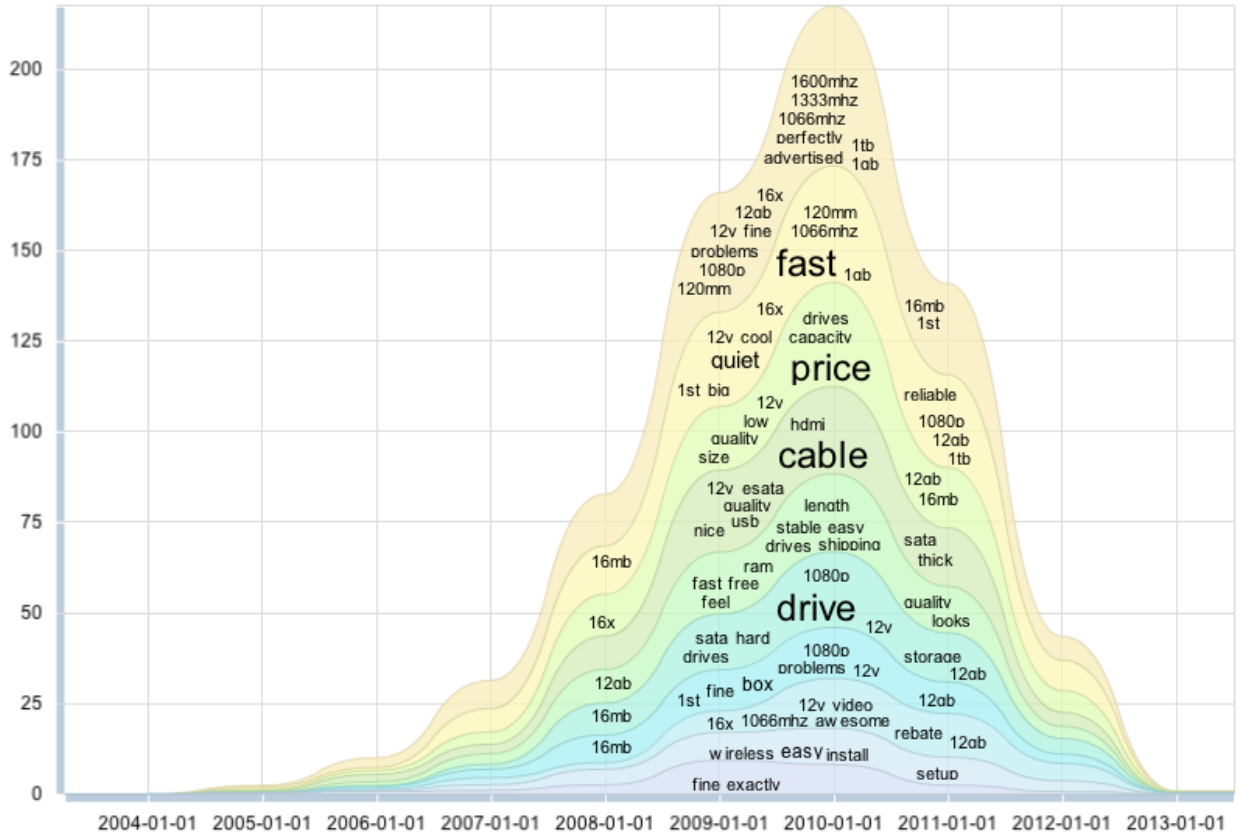

Figure 11: Newegg 12 reviews dataset with NMI=0.23, $\lambda = 87$, $k = 15$.

5.3 Conclusion and Directions for Further Work

The most important result of this work is that NMI is a good measure of quality of a clustering solution. The Newegg clusterings that had low NMI compared to the TDT2 clusterings intuitively appear to be worse clusterings. To further emphasize this point, we consider a clustering of Newegg data where the W matrix was constructed by simply setting each document to the class it belongs in, and then using RRI to find the appropriate H matrix; such a clustering has NMI=1 and is presented in Figure 12. This solution has a much higher NMI value than the solutions presented

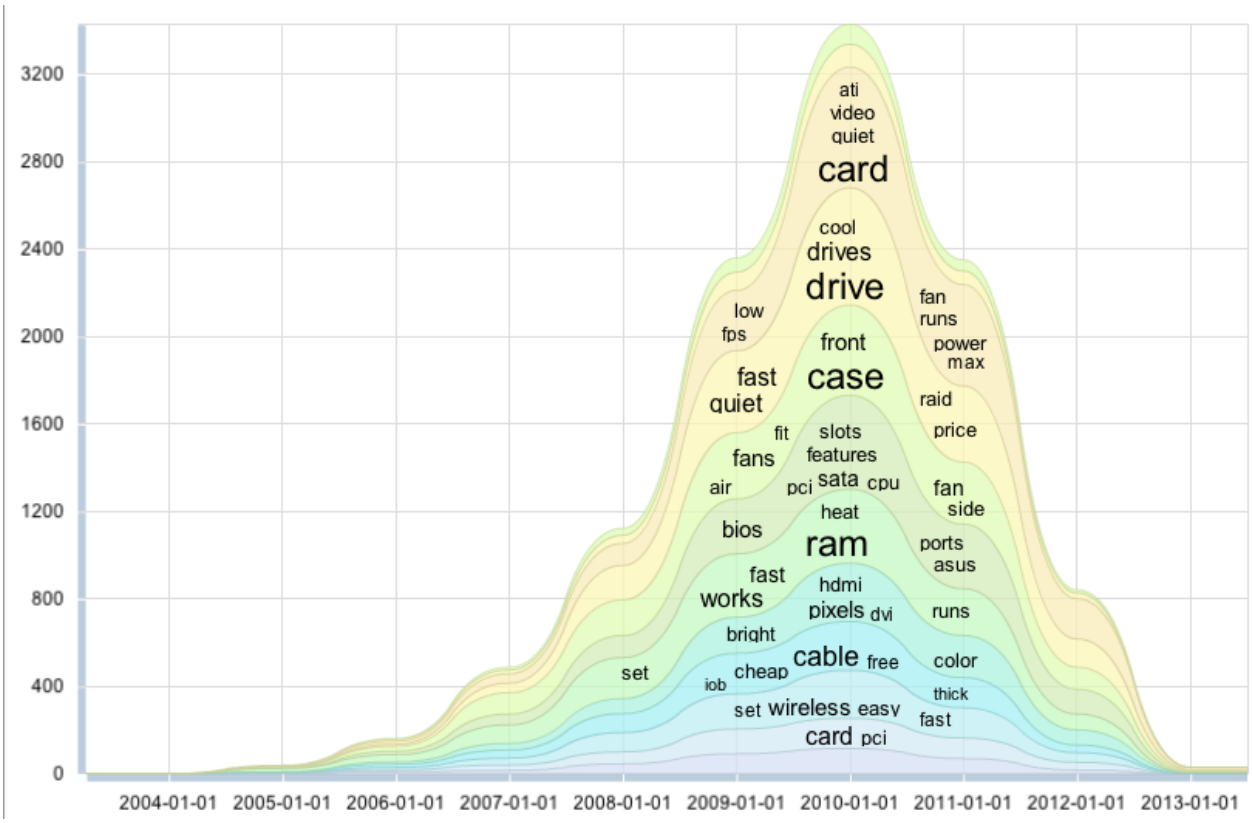

Figure 12: Newegg 12 reviews dataset with NMI=1, λ =?, $k = 12$.

in Figure 11, and is intuitively much better: each cluster has a clear set of defining words, and it is clear how each cluster corresponds to each of the original product categories.

From this result two important questions remain as directions for future work. First, both the Newegg and TDT2 datasets were approximately the same size, i.e. approximately ten thousand documents and a dictionary less than ten thousand words large, however it was possible to achieve much higher NMI solutions for the TDT2 dataset than it is for the Newegg dataset. Furthermore

selecting optimal (λ, k) had a larger effect on the TDT2 clusterings than it did on the Newegg clusterings. These facts point that perhaps there is another important parameter, or more likely a set of parameters, that determine how RRI works, and as a result affect the NMI of different clustering solutions. Further work can be done in a similar fashion as this project was done to determine how NMI is affected by for example smoothness of the W, H matrices, or more generally on what kinds of results a particular non-negative matrix factorization technique is able to achieve.

A Derivative of NMI with respect to λ

Here we consider the difficulties of determining a formal dependence of NMI on λ. To do this we rephrase Equation 15 in terms the output of our clustering algorithm. Let $W(m,n)$ denote the nth element in the mth row of a clustering solution matrix $W \in \mathbf{R}^{n\times k}$. Let $W(:,n)$ denote the nth column, and $W(m,:)$ denote the mth row of W. Let $\mathbf{I}_i$ be an indicator vector such that $\mathbf{I}_i(m) = 1$ if the mth document is classified as belonging to the ith class and 0 otherwise. Then Equation 15 can be rewritten as

$$\mathrm{NMI}(W,I) = \frac{2\sum_{r=1}^{k}\sum_{i=1}^{q}\frac{||W(:,r)^T\mathbf{I}_i||_1}{n}\log_b\frac{||W(:,r)^T\mathbf{I}_i||_1}{||W(:,r)||_1||\mathbf{I}_i||_1}}{\sum_{i=1}^{q}\frac{-||\mathbf{I}_i||_1}{n}\log_b\frac{-||\mathbf{I}_i||_1}{n}+\sum_{r=1}^{k}\frac{-||\mathbf{W}(:,r)||_1}{n}\log_b\frac{-||\mathbf{W}(:,r)||_1}{n}}. \tag{22}$$

Formally we have

$$\frac{\partial \mathrm{NMI}}{\partial\lambda} = \frac{\partial \mathrm{NMI}}{\partial W}\frac{\partial W}{\partial\lambda} + \frac{\partial \mathrm{NMI}}{\partial I}\frac{\partial I}{\partial\lambda}, \tag{23}$$

but since $\frac{\partial I}{\partial\lambda} = 0$, the second term disappears. To bring the expression closer to RRI NMI, we note that

$$\frac{\partial \mathrm{NMI}}{\partial W}\frac{\partial W}{\partial\lambda} = \sum_{r=1}^{k}\frac{\partial \mathrm{NMI}}{\partial W}\frac{\partial W}{\partial||W(:,r)||_1}\frac{\partial||W(:,r)||_1}{\partial\lambda} \tag{24}$$

From Equation 22 it is apparent that the first two factors of the derivative will not contain λ, and since we are interested in the dependence of λ we simply let these terms be some functions so that

$$\sum_{r=1}^{k}\frac{\partial \mathrm{NMI}}{\partial W}\frac{\partial W}{\partial||W(:,r)||_1}\frac{\partial||W(:,r)||_1}{\partial\lambda} = \sum_{r=1}^{k} f_1(W)f_2(||W(:,r)||_1)\frac{\partial||W(:,r)||_1}{\partial\lambda} \tag{25}$$

From Equation 2 for a particular $r = t$

$$||W(:,r)||_1 = \sum_{i=1}^{n}\frac{R_t(i,:)H(:,r)-\lambda}{||H(:,r)||_2^2} \tag{26}$$

where $R_t = X - \sum_{i\neq t}^{k} W(:,i)H(:,i)^T$ is defined as in Section 2.5.1. It is difficult to take the partial derivative because

$$||H(:,r)||_1 = \sum_{i=1}^{n}\frac{R_t(i,:)^TW(:,r)-\lambda}{||W(:,r)||_2^2}, \tag{27}$$

so that the relationship of W and λ seems obscured. Rather than attempting to continue, a statistical approach is more expedient for capturing the relationship between NMI and λ.

B Matlab code for λ, k finding algorithm and for RRI

```
function [ nmis, lambdas, epsilons, Ps] = optLambdaKInterpol( X,classes,k_init )
%optLambdaKInterpol Uses polynomial sampling to find optimal lambda and k
%   See section 4.2

poly_deg    =3; %the polynomial degree we use for interpolation, 3 for cubic

k_range     =4; %the m of step 1. of Section 4.2
if k_init ≥ 2+k_range
    kmin    =k_init-k_range+1;
    kmax    =k_init+k_range;
else
    kmin    =2;
    kmax    =k_init+k_range;
end
minlambda   =0;
maxlambda   =max(sum(X));

K           =kmin:kmax;
nmis        = cell(kmax,1); %we store the results of nmis for
epsilons    = cell(kmax,1); %the confidence interval width
lambdas     = cell(kmax,1);
alpha       =0.05;          %the confidence of the intervals we use to compare ...
    different lambda_k will be 1-alpha
confidence_overlap = 0.50;
conf        = @(x) ...
    tinv(1-alpha/2,max(size(x,1),size(x,2))-1)*sdev_to_max(x)/sqrt(max(size(x,1),size(x,2)));

numRuns     =2;
b           =maxlambda;
a           =minlambda;
cheb        = @(i) (b+a)/2 +(b-a)*cos((2*i+1)*pi/(2*poly_deg+2))/2;
L           = cheb(0:poly_deg);

for k=K
    %sample the chebyshev lambdas numRun number of times for each k
    nmi=zeros(size(L,2),numRuns);
    for run=1:10 %gather 10 points for the polynomial fitting for each of the ...
        chebyshev lambdas
        for l=L
            [Winit,Hinit]=spinitialize_nmf(X,k,l);
            [W,¬,¬,¬,¬]=rri(X, Winit, Hinit, l, 1);
            W = (spdiags (sum (abs(W),2), 0, size(X,1), size(X,1)) \ W); %column ...
                normalize W
            nmi(find(L==l),run)=NMI(W,classes);
        end
    end
    [Ps{k},S]=polyfit(L,max(nmi'),3); %fit a polynomial to the samples
    der = polyder(Ps{k});
    crit=roots(der);
    lambda_candidates=sort([0;crit(find(crit>0))])'; %0 and the positive critical ...
        points of the polynomial
    nmis{k}=zeros(numRuns,size(lambda_candidates,2));
    for run=1:numRuns
        for l=lambda_candidates
```

```
50              [Winit,Hinit]=spinitialize_nmf(X,k,l);
51              [W,¬,¬,¬,¬]=rri(X, Winit, Hinit, l, 1);
52              W = (spdiags (sum (abs(W),2), 0, size(X,1), size(X,1)) \ W); %column ...
                    normalize W
53              nmis{k}(run,find(lambda_candidates==l))=NMI(W,classes);
54          end
55      end
56      lambdas{k}=lambda_candidates;
57      epsilons{k}=conf(nmis{k});
58  end
59
60  terminate = 0;
61  while ¬terminate
62      maxs=cellfun(@max,nmis','UniformOutput',false);
63      min_epsilon_k=-1;
64      max_mean_k=-1;
65      max_mean=0;
66      min_epsilon=1;
67      for k=K
68          epsilons{k}=conf(nmis{k});
69          max_this_k = max(maxs{k});
70          if max_this_k > max_mean
71              max_mean_k=k;
72              max_mean=max_this_k;
73          end
74      end
75
76      absolute_overlap=0;
77      conf_min=max_mean-epsilons{max_mean_k}(find(maxs{max_mean_k}==max_mean)); %the ...
            min & max confidence bounds for the best solution
78      conf_max=max_mean+epsilons{max_mean_k}(find(maxs{max_mean_k}==max_mean));
79      for k=K
80          for m=maxs{k}
81              overlap_m=0;
82              if m ≠ max_mean
83                  if m < conf_min
84                     overlap_m = max(0,m+epsilons{k}(find(maxs{k}==m))-conf_min);
85                  end
86                  if m > conf_min
87                     overlap_m = min(m-conf_min,epsilons{k}(find(maxs{k}==m)))+ ... ...
                           %bottom part
88                                    min(conf_max-m,epsilons{k}(find(maxs{k}==m))); ...
                                              %top part
89                  end
90              end
91              if overlap_m > absolute_overlap
92                  absolute_overlap=overlap_m;
93              end
94          end
95      end
96
97      % if the highest mean has also the smallest confidence interval, we are
98      % done
99      ratio=absolute_overlap/(conf_max-conf_min);
100     fprintf(1,'max_mean=%f k=%d absolute_overlap=%f 2e=%f ...
            ratio=%f\n',max_mean,max_mean_k,absolute_overlap,(conf_max-conf_min),ratio);
101     if ratio ≤ confidence_overlap;
102         terminate=1;
```

```
103         fprintf(1,'terminating with k=%d lambda=%f numruns ...
                done=%d\n',max_mean_k,lambdas{k}(find(max(nmis{k})==max_mean)),numRuns);
104     else
105         fprintf(1,'doing another run\n');
106         numRuns=numRuns+1;
107         parfor k=kmin:kmax
108             for l=lambdas{k}
109                 [Winit,Hinit]=spinitialize_nmf(X,k,l);
110                 [W,~,~,~,~]=rri(X, Winit, Hinit, l, 1);
111                 W = (spdiags (sum (abs(W),2), 0, size(X,1), size(X,1)) \ W); ...
                        %column normalize W
112                 nmis{k}(numRuns,find(lambdas{k}==l))=NMI(W,classes);
113                 %fprintf(1,'\t[%d][l=%f] nmi=%f]\n',run,l,nmi_res);
114             end
115             epsilons{k}=conf(nmis{k});
116         end
117
118     end
119 end
120 end
```

```
function [W,H,fro,wOneNorm,obj] = rri(V, W, H, ...
    lambda,z,tol,timelimit,maxiter,epsilon, fixWH, constraints )
%% [W,H,fro,wOneNorm,obj] = rri(V, W, H, lambda,tol,timelimit,maxiter,epsilon, ...
    fixWH, constraints,z )
% Rank-one residue iterations for NMFs to solve
%   argmin_{W ≥0, H ≥0} 0.5*|| V - W H ||^2 + epsilon*(||W||^2 + ||H||^2)   + ...
    lambda*||W||_0/N
%
% V: n x d data matrix (documents-words for example)
% W : n x k initial document-topic matrix
% H : k x d initial topic-word matrix
% tol : convergence tolerance
% timelimit : return a solution within these many seconds
% maxiter : maximum number of iterations
% epsilon : regularization parameter
%
% fixWH=[1 0] fixes W and only learns H
% fixWH=[0 1] fixes H and only learns W
%
% Defaults are ...
    tol=0.001,timelimit=50,maxiter=100,epsilon=1e-12,fix[WH]=[0,0],lambda=0,constraints='simplex'.
%
% Author: Vikas Sindhwani (vsindhw@us.ibm.com)
% 2010

warnMe=0;
if nargin<11
    constraints='simplex';
    if nargin<10
        fixWH=[0 0];
        if nargin<9
            epsilon=1e-12;
            if nargin<8
                maxiter=100;
                if nargin<7
                    timelimit=50;
                    if nargin<6
                        tol=0.001;
                        if nargin<5
                            z=1;
                            if nargin<4
                                lambda=0;
                            end
                        end
                    end
                end
            end
        end
    end
end


if ¬strcmp(constraints,'simplex')  &&  ¬strcmp(constraints,'nnl2')
    disp('No constraint on H selected - expect problems')
end

[N,D]=size(V);
K=size(H,1);

```

```
lambdaOverN=lambda/N;

obj=Inf;
start=cputime;

normV = norm(V,'fro')^2;

Δ=0;

iter=0;
terminate = 0;

% work with H' for efficiency
H = H';

while ¬terminate

    if iter>maxiter
        terminate=1;
        continue;
    end


    if fixWH(1)==0

        obj2 =0.0;
        obj3a=0.0;
        obj3b=0.0;
        obj4 = 0.0;
        obj5 = 0.0;


        for r = 1:K

            % what we are optimizing in this round
            %h = H(r,:)';
            h = H(:,r);

            %Hh = H*h;
            Hh = H'*h;

            Hh(r)=0;

            Vh = V*h;


            Rh = max(Vh - W*Hh - lambdaOverN,Δ);

            norm_h2 = h'*h;

            if norm_h2>0
                w = Rh/norm_h2;
            else
                w = zeros(length(Rh),1);
            end

            W(:,r) = w;


```

```
115
116             wW = w'*W;
117
118             obj2  = obj2 + w'*Vh;
119             %obj3a = obj3a + w'*W(:,1:r-1)*Hh(1:r-1);
120             obj3a = obj3a + wW(1:r-1)*Hh(1:r-1);
121             obj3b = obj3b + (h'*h)*(w'*w);
122             obj4 = obj4 + (h'*h) + (w'*w);
123             sumw = sum(w);
124             obj5 = obj5 + sumw;
125         end
126
127     end
128
129     % update in order to alter lambda in the next step
130     %nonnegavg = obj5/sum(sum(W==0));
131
132     %%%%%%%%%%%%%%%%% CONVERGENCE CHECK %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
133
134     prev_obj=obj;
135
136     %obj_original = normV - 2*trace(H*V'*W) + trace((H*H')*(W'*W)) + ...
            epsilon*(norm(W,'fro')^2 + norm(H,'fro')^2);
137
138     if (fixWH(1)==1) && iter==0
139
140         obj = normV - 2*trace(H'*V'*W) + trace((H'*H)*(W'*W)) + ...
                epsilon*(norm(W,'fro')^2 + norm(H,'fro')^2) + lambdaOverN*sum(sum(W));
141
142     else
143
144         obj = normV - 2*obj2 + (2*obj3a +obj3b) + epsilon*obj4 + lambdaOverN*obj5; ...
                % last term uses definition of trace + symmetry
145
146     end
147
148     iter=iter + 1;
149     elapsed=cputime-start;
150     if prev_obj<obj && warnMe
151
152         fprintf('obj function increased - Something is wring.\n  Information on ...
                this run stored in Problem.mat\n');
153         probX = V;
154         probWinit=W;
155         probHinit=H';
156         probLam=lambda;
157         save('Problem', 'probX','probWinit','probHinit','probLam');
158     end
159
160     if abs(prev_obj-obj)<prev_obj*tol || (elapsed>timelimit)
161         terminate=1;
162     end
163
164
165     %%%%%%%%%%%%%%%%%%%%%%%%%% CONVERGENCE CHECK END %%%%%%%%%%%%%%%%%%%%%%%%%% ...
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
166
167     if fixWH(2)==0
168
```

```
169         obj2=0.0;
170         obj3a=0.0;
171         obj3b=0.0;
172         obj4 = 0.0;
173         obj5 = 0.0;
174         for r=1:K
175
176             % update equations for h
177             w = W(:,r);
178             Ww = W'*w;
179             Ww(r)=0;
180             Vw = V'*w;
181
182             norm_w2 = w'*w;
183
184             if norm_w2>0
185                 q = (Vw - H*Ww)/(w'*w);
186             else
187                 q = zeros(length(Vw),1);
188             end
189
190
191             switch constraints
192                 case 'nnl2'
193                     q_plus = max(q,\Delta);
194                     h = q_plus/max(sqrt(q_plus'*q_plus),1);
195                 case 'simplex'
196                     h = simplex_projection(q,z);
197                 otherwise
198                     h=q;
199             end
200
201
202
203             H(:,r) = h;
204
205             Hh = H'*h;
206
207             obj2  = obj2 + h'*Vw;
208             %obj3a = obj3a + h'*H(1:r-1,:)'*Ww(1:r-1); % using the fact that ...
                    (H*H') is symmetric.
209             obj3a = obj3a + Hh(1:r-1)'*Ww(1:r-1);
210             obj3b = obj3b + (h'*h)*(w'*w);
211             obj4 = obj4 + (h'*h) + (w'*w);
212             obj5 = obj5 + sum(w);
213         end
214
215     end
216     fro = normV - 2*obj2 + (2*obj3a +obj3b);
217 end
218
219 H = H';
220 wOneNorm = sum(sum(W));
```

References

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. 1st ed. 2006. Corr. 2nd printing. Springer, Oct. 2007. ISBN: 0387310738. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0387310738.

[2] David M. Blei et al. "Latent dirichlet allocation". In: *Journal of Machine Learning Research* 3 (2003), p. 2003.

[3] Kenneth Ward Church and Patrick Hanks. "Word association norms, mutual information, and lexicography". In: *Comput. Linguist.* 16 (1 1990), pp. 22–29. ISSN: 0891-2017. URL: http://portal.acm.org/citation.cfm?id=89086.89095.

[4] Marc Damashek. "Gauging Similarity with n-Grams: Language-Independent Categorization of Text". In: *Science* 267.5199 (1995), pp. 843–848. DOI: 10.1126/science.267.5199.843. eprint: http://www.sciencemag.org/content/267/5199/843.full.pdf. URL: http://www.sciencemag.org/content/267/5199/843.abstract.

[5] Scott Deerwester et al. "Indexing by latent semantic analysis". In: *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE* 41.6 (1990), pp. 391–407.

[6] Chris Ding, Tao Li, and Wei Peng. "NMF and PLSI: equivalence and a hybrid algorithm". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. SIGIR '06. Seattle, Washington, USA: ACM, 2006, pp. 641–642. ISBN: 1-59593-369-7. DOI: http://doi.acm.org/10.1145/1148170.1148295. URL: http://doi.acm.org/10.1145/1148170.1148295.

[7] John Duchi, Yoram Singer, and Tushar Chandra. *Efficient Projections onto the 1-Ball for Learning in High Dimensions*. 2008.

[8] Ngoc-Diep Ho. "Nonnegative Matrix Factorizations Algorithms and Applications". PhD thesis. Boston, MA, USA: Académie universitaire Louvain, 2008.

[9] Thomas Hofmann. "Probabilistic Latent Semantic Indexing". In: *SIGIR*. 1999, pp. 50–57.

[10] Daniel D. Lee and H. Sebastian Seung. "Algorithms for Non-negative Matrix Factorization". In: *In NIPS*. MIT Press, 2001, pp. 556–562.

[11] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN: 0521865719, 9780521865715.

[12] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. MIT Press, 2001, pp. 849–856.

[13] William M. Rand. "Objective Criteria for the Evaluation of Clustering Methods". English. In: *Journal of the American Statistical Association* 66.336 (1971), pp. 846–850. ISSN: 01621459. URL: http://www.jstor.org/stable/2284239.

[14] M. Rosell, V. Kann, and J. E. Litton. "Comparing comparisons: Document clustering evaluation using two manual classifications". In: *International Conference on Natural Language Processing, Allied Publishers Private Limited, pp.* Citeseer. 2004, pp. 207–216.

[15] C. E. Shannon. "A Mathematical Theory of Communication". In: *Bell Sys. Tech. J.* 27 (1948), pp. 379–423, 623–656.

[16] Ying Zhao, George Karypis, and Usama Fayyad. "Hierarchical Clustering Algorithms for Document Datasets". In: *Data Mining and Knowledge Discovery* 10 (2 2005). 10.1007/s10618-005-0361-3, pp. 141–168. ISSN: 1384-5810. URL: http://dx.doi.org/10.1007/s10618-005-0361-3.