Novel Methods in Information Retrieval

Vahed Qazvinian Dragomir R. Radev
School of Information and
Department of EECS
University of Michigan
Winter, 2008
{vahed,radev}@umich.edu

Contents

1 Introduction

Information retrieval (IR) is the science of information search within documents, relational databases, and the World Wide Web (WWW). In this work, we have tried to review some novel methods in IR theory. This report covers a number of the state of art methods in a wide range of topics with a focus on graph-based techniques in IR.

This report is created based on the literature review done as a requirement of the Directed Study, EECS 599, course at the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. The first author would like to thank Prof. Dragomir R. Radev for his discussions, reviews, and useful suggestions to support this work.

2 Random Walks

2.1 Preliminaries

A random walk is a mathematical formalization of successive movements in random directions. This analysis applies to Computer Sciences, Physics, Theory of Probability, Economics and some other fields of studies. In general, the position of a random walker is determined by its previous state (position) and a random variable that determines the subsequent step length and direction. More formally,

$$X(t+\tau) = X(t) + \Phi(\tau)$$

where $X(t)$ is the position of the random walker at time t and $\Phi(\tau)$ is a random variable that determines the rule to take the next step.

Different categorizations for random walks have been proposed, based on whether they are discrete or continues, biased or unbiased, one dimensional or of higher dimensions. The simplest form of a random walk is a discrete one-dimensional walk in which $\Phi(\tau)$ is a Bernoulli random variable with p being the probability of a positive direction, and $1-p$ being the probability of a negative direction. Polya's theorem [63] indicates that a random walker on an infinite lattice in d-dimensional space is bound to return to the starting point when $d=2$, but it has a positive probability of escaping to infinity without returning to the starting point when $d \geqslant 3$. The walk that will meet its starting place is called *recurrent*, while if there's a positive probability of escaping it is called *transient*.

An infinite lattice can be considered as an extremely big graph that fits in it. Let G_r be the subgraph of an infinite lattice in which no node has a *lattice distance* of greater than r from the origin. This means that no shortest path along the edges of the lattice with its head in origin has length greater than r. Let $\partial G(r)$ be the *sphere* of radius r about the origin, (points with the exact distance r from the origin). $\partial G(r)$ looks like a square in a 2-dimensional lattice. Now consider a random walker that starts its walk from origin. If $p_{esc}^{(r)}$ be the probability that the random walker reaches $\partial G(r)$ before returning to origin, then the *escape probability* of the random walker is $\lim_{r\to\infty} p_{esc}^{(r)}$ which decreases as r increases. With this definition, If this limit is 0, the walk is recurrent, otherwise it is transient.

In the first part of this chapter we will have an overview of the electrical approach to proof Polya's theorem discussed in [28], and in the second part we discuss the work described by Aldous and Fill [5].

2.2 Random Walks and Electrical Circuits

One of the attempts has been to interpret the Polya's theorem as a statement about electric networks, and then to prove the theorem using the electrical theoretic point of view [28].

To determine $p_{esc}^{(r)}$ electrically, the authors in [28] ground all the points of $\partial G(r)$ and maintain the origin at one volt, and measure the current $i(r)$ flowing into the circuit. They show that

$$p_{esc}^{(r)} = \frac{i(r)}{2d}$$

For d being the dimension of the lattice. Given that the voltage is 1, then $i(r)$ is the effective conductance between the origin and $\partial G(r)$:

$$i(r) = \frac{1}{R_{eff}^{(r)}}$$

So

$$p_{esc}^{(r)} = \frac{1}{2dR_{eff}^{(r)}}$$

If the effective resistance from origin to infinity is R_{eff} then

$$p_{esc} = \frac{1}{2dR_{eff}}$$

Thus if the effective resistance of the d-dimensional lattice is infinite, then the escape probability is zero and the walk is recurrent. In other words: "simple random walk on a graph is recurrent if and only if a corresponding network of 1-ohm resistors has innite resistance *out to infinity*". It is trivial that an innite line of resistors has innite resistance. Using this fact, Doyle and Snell [28] conclude that simple random walk on the 1-dimensional lattice is recurrent, which confirms Polyas theorem.

For the two-dimensional case, they use the *Shorting Law* to calculate the resistance of the lattice network out to infinity as

$$\sum_{n=1}^{\infty} \frac{1}{4n+8}$$

which tends to ∞. So in the two dimensional case:

$$p_{esc} = \frac{1}{2dR_{eff}} = \frac{1}{2d\sum_{n=1}^{\infty}\frac{1}{4n+8}} = 0$$

Which again confirms that simple random walk on the 2-dimensional lattice is recurrent as stated by Polyas theorem.

For three dimensional lattice, denote by $P(a,b,c;n)$ the probability that the random walker be at (a,b,c) at time n.

$$p(0,0,0;0) = 1$$

and

$$\begin{aligned} p(a,b,c;n) &= \frac{1}{6}p(a-1,b,c;n-1) + \frac{1}{6}p(a+1,b,c;n-1) \\ &+ \frac{1}{6}p(a,b-1,c;n-1) + \frac{1}{6}p(a,b+1,c;n-1) \\ &+ \frac{1}{6}p(a,b,c-1;n-1) + \frac{1}{6}p(a,b,c+1;n-1) \end{aligned}$$

Using the technique of generating functions they reach

$$\begin{aligned} p(a,b,c;n) &= \frac{1}{(2\pi)3}\cdot \\ & \int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\left(\frac{\cos x + \cos y + \cos z}{3}\right)^n \cos(xa)\cos(yb)\cos(zc)dxdydz \end{aligned}$$

Then the expected value of the number of returns to origin is reached by assigning $a = b = c = 0$ and summing over n:

$$m = \frac{3}{(2\pi)^3} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \frac{1}{3 - (cosx + cosy + cosz)} dxdydz$$

A simple solution to this integral is proposed in [34] who evaluated this integral in terms of gamma functions:

$$m = \frac{\sqrt{6}}{32\pi^3} \Gamma(\frac{1}{24}) \Gamma(\frac{5}{24}) \Gamma(\frac{7}{24}) \Gamma(\frac{11}{24}) = 1.516386...$$

where

$$\Gamma(x) = \int_0^{\infty} e^{-t} t^{x-1} dt$$

If u is the probability that a random walker, starting at the origin, will return to the origin, then the probability that the walker will be there exactly k times (counting the initial time) is $u^k(1 - u)$. Thus, if m is the expected number of times at which the walker is at the origin:

$$m = \sum_{k=1}^{\infty} k u^{k-1} (1 - u) = \frac{1}{1 - u}$$

So,

$$u = \frac{m - 1}{m}$$

This shows, in this particular 3-dimensional lattice, that the probability of a random walker, starting at the origin, returning to the origin is

$$u = 0.340537...$$

Which means that in the 3-dimensional lattice there is a non-zero probability that the random walker will not return to the origin and will *escape*.

2.3 Random Walks on Graphs

In this section, we mainly focus on the graph application of Markov chains [5].

A Markov Chain is a stochastic process $\{X_n, n = 0, 1, 2, \cdots\}$ that takes on a finite or countable number of possible values. Whenever the process is in state i, there's a fixed probability P_{ij} that it will in state j in the next step.

$$P\{X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \cdots, X_1 = i_1, X_0 = i_0\} = P_{ij}$$

where

$$P_{ij} \geqslant 0,$$

$$i, j \geqslant 0,$$

$$\sum_{j=0}^{\infty} P_{ij} = 1$$

P is called the one-step transition Probability matrix.

$$P = \begin{pmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ \vdots & \vdots & \vdots & \\ P_{i0} & P_{i1} & P_{i2} & \cdots \\ \vdots & \vdots & \vdots & \end{pmatrix}$$

If π is the stationary distribution of a finite irreducible discrete-time chain (X_t), the chain is *reversible* if

$$\pi_i p_{ij} = \pi_j p_{ji}$$

In fact, if (X_t) is the stationary chain (X_0 has distribution π) then

$$(X_0, X_1, \cdots, X_t) = (X_t, X_{t-1}, \cdots, X_0)$$

In the forth section, the authors talk about coupling which is a methodology to find an upper-bound for $\bar{d}(t) := \max_{ij} \|P_i(X_t = .) - P_j(X_t = .)\|$. Coupling is a joint process $((X_t^{(i)}, X_t^{(j)}), t \geqslant 0)$ such that: $X_t^{(i)}$ is distributed as the chain started at i, and $X_t^{(j)}$ is distributed as the chain started at j. Then the assumption is made that there is a random time $T^{ij} < \infty$ such that $X_t^{(j)} = X_t^{(j)}, T^{ij} \leqslant t < \infty$ and T^{ij} is called the *coupling time.* The coupling inequality is

$$\begin{aligned} \|P_i(X_t \in .) - P_j(X_t \in .)\| &= \|P_i(X_t^{(i)}) \in .) - P_j(X_t^{(j)}) \in .)\| \\ &\leqslant P(X_t^{(i)} \neq X_t^{(j)} \\ &\leqslant P(T^{ij} > t) \end{aligned}$$

The calculation of random walks on large graphs can be done under two settings. First, when the graph is just 1-dimensional. Second, when the graph is highly symmetric. Aldous and Fill [5] further talk about tree states, and that on a n-vertex tree, a random walk's stationary distribution is

$$\pi_v = \frac{r_v}{2(n-1)}$$

3 Semi-supervised Learning

3.1 Survey

Many semi-supervised learning methods have been used before. Some are EM with generative mixture models, self-training, co-training, transductive support vector machines, and graph-based methods. There is no direct answer which one should be used or which is the best. The reason is that the labeled data is scarce and semi-supervised learning methods tend to make strong model assumption which is highly dependent on the problem structure.

The big picture is that semi-supervised learning methods use unlabeled data to modify the hypotheses ($p(y|x)$) derived from labeled data ($p(x)$) alone. In this section semi-supervised learning refers to semi-supervised classification, in which one has additional unlabeled data and the goal is classification, unless otherwise noted.

3.1.1 Co-training

Co-training [55] is a semi-supervised learning techniques which makes three preliminary assumptions:

1. The feature space can be splitted up into two separate sets.
2. Each feature space is sufficient to train a good classifier.
3. The two sets are conditionally independent given the class.

Initially, each classifier is trained using the initial labeled data on the corresponding feature space.Then both classifiers classify all unlabeled data. In each iteration each classifier adds n negative data points and p positive ones, about which it is the most confident to the labeled data and returns the rest to the shared unlabeled data. Both classifiers will then re-train using the new labeled data.

Input::

- $F = (F_1, F_2, ., F_m)$ which are m semantically different feature sets (that can be considered as different views of the data)
- $C = (C_1, C_2, ., C_m)$ which are m supervised learning classifiers (each of which corresponds to a unique feature set)
- L is a set of labeled training data
- U is a set of unlabeled training data

The procedure of Co-training is as follows.

- Training each classifier C_i initially on L with respect to F_i
- For each classifier C_i:
  - C_i labels the unlabeled training data from U based on F_i
  - C_i chooses the top p positive and top n negative labeled examples E from U according to the confidence of the prediction

  - Remove E from U
  - Add E into L with corresponding labels predicted by C_i

More generally, we can define learning paradigms that use the agreement among different learners in which multiple hypotheses are trained from the shared labeled data, and are necessary for making predictions on given unlabeled data. In general, multiview learning models do not require the assumptions made by Co-Training.

3.1.2 Graph-based Methods

In graph based methods, the techniques are applied on graphs with labeled an unlabeled nodes, and similarity of nodes as edges. Graph based methods are problems that deal with estimating a function f on the graph where f should satisfy two properties:

1. Should be close to the given labels y_L on the labeled nodes. (loss function)
2. Should be smooth on the whole graph. (regularizer)

Most of the graph based methods are only different from one another in that they have different loss functions and regularizers. It is more important to construct a good graph than to choose among different methods. We will itemize some of the methods and discuss the graph construction after that.

- **Mincut**
  Blum and Chawla [14] represent the problem as a mincut problem in which positive labels act as sources and negative labels act as sinks. The goal is to find a minimum set of edges whose removal blocks all flow from the sources to the sinks. After that, all nodes connected to the sources are labeled as positive, and all nodes connected to the sinks are labeled as negative. The loss function is a quadratic loss with infinity weight

$$\infty \sum_{i \in L} (y_i - y_{i|L})^2$$

  This ensures that the values on the labeled data are fixed at their given labels. The regularizer is

$$\frac{1}{2} \sum_{i,j} \omega_{ij} |y_i - y_j| = \frac{1}{2} \sum_{i,j} \omega_{ij} (y_i - y_j)^2$$

  The above equality holds since y takes binary values. Putting all together, the goal of the mincut method is to minimize

$$\infty \sum_{i \in L} (y_i - y_{i|L})^2 + \frac{1}{2} \sum_{i,j} \omega_{ij} (y_i - y_j)^2$$

  given the constraint

$$y_i \in \{0, 1\}, \forall i$$

- **Gaussian Random Fields**
  Zhu et, al [74] introduce the Gaussian random fields and harmonic function methods which is a relaxation of discrete Markov random fields. This can be viewed as having a quadratic loss function with infinity weight, so that the labeled data are fixed at given label values, and a regularizer based on the graph combinatorial Laplacian (δ)

$$\begin{aligned} & \infty \sum_{i \in L} (f_i - y_i)^2 + \frac{1}{2} \sum_{i,j} \omega_{ij} (f_i - f_j)^2 \\ = & \infty \sum_{i \in :} (f_i - y_i)^2 + f^T \Delta f \end{aligned}$$

  $f_i \in R$ is the key relaxation to Mincut.

- **Local and Global Consistency**
  The method of local and global consistency [76] uses the loss function

$$\sum_{i=1}^{n} (f_i - y_i)^2$$

  But in the regularizer it uses a *normalized Laplacian*, that is, $D^{-1/2} \Delta D^{-1/2}$. The regularizer is

$$\frac{1}{2} \sum_{ij} \omega_{ij} (\frac{f_i}{\sqrt{D_{jj}}} - \frac{f_j}{\sqrt{D_{jj}}})^2 = f^T D^{-1/2} \Delta D^{-1/2} f$$

- **Local Learning Regularization**
  The solution of graph-based methods can often be viewed as local averaging. In other words, the solution $f(x_i)$ in an unlabeled point x_i is the weighted average of its neighbors' solutions. Since most of the nodes are unlabeled, we do not require the solution to be exactly equal to the local average, but regularize f so they are close. A more general approach is to generalize local averaging to a local linear fit. To do so, one should build a local linear model from x_i's neighbors to predict the value at x_i. Thus, the solution $f(x_i)$ is regularized to be close to this predicted value.

- **Tree-Based Bayes**
  In this method, a tree, T, is constructed with the labeled and unlabeled data as the leaf nodes, and has a mutation process. In the *mutation* process a label at the root propagates down to the leaves. While moving down along edges, a label mutates at a constant rate. This makes the tree uniquely define the probabilistic distribution $P(X|Y)$ on discrete Y labelling. In fact, if two leaf nodes are closer in the tree, they have a higher probability of sharing the same label.

  Although the graph is the core of the graph-based methods, its construction for this purpose has not been studied widely. Majer and Hein [47] propose a method to denoise point samples from a manifold. This can be considered as a preprocessing step to build a better graph in order to do the semi-supervised learning on the graph which is less noisy. Such preprocessing results in a better semi-supervised classification accuracy.

3.2 Semi-supervised Classification with Random Walks

In this section we reviewed a method of semi-supervised classification using random walks described in [68]. In their work, Szummer and Jaakkola [68] first create a weighted K-nearest neighbor graph, with a given metric, and form the one-step transition probability matrix as

$$p_{ik} = \frac{\omega_{ik}}{\sum_j \omega_{ij}}$$

if i and k are neighbor nodes, otherwise $p_{ik} = 0$.

Two points are considered close if they have nearly the same distribution over the starting states. When $t \to \infty$ all the points are indistinguishable if the original neighbor graph is connected. If P is the one step transition probability then P^t is the t-step transition probability which is row-stochastic, i.e. rows sum to one. A point k, whether labeled or unlabeled, is interpreted as a sample from the t-step Markov random walk. The posterior probability o the label for point k is given by

$$P_{post}(y|k) = \sum_i P(y|i) P^t_{ik}$$

The class is then the one that maximizes the posterior probability:

$$c_k = argmax_c \{P_{post}(y = c|k)\}$$

The authors then use two separate techniques to estimate the unknown parameter $P(y|i)$: Maximum likelihood with EM and maximum margin given constraints. They conclude that the Markov random walk representation provides a robust approach to classify datasets with significant manifold structure and very few labeled data.

3.3 Graph-Based Semi-supervised learning with Harmonic Functions

Most of the semi-supervised learning algorithms are based on the manifold structure and the assumption that similar examples should belong to the same class. The goal in graph classification is to learn a function $f : V \to R$. For all labeled nodes v_i we have $f(v_i) \in \{0, 1\}$.
The desired situation is that close and similar nodes are assigned to the same classes. Therefore, an *Energy function* is defined as:

$$E(f) = \frac{1}{2} \sum_{i,j} \omega_{ij} (f(i) - f(j))^2$$

Solving the classification problem, now reduces to minimizing the energy function. A solution f to this minimization problem is a *harmonic* function with the following property:

$$f(j) = \frac{1}{d_j} \sum_{i \sim j} \omega_{ij} f(i)$$

for $j = l + 1, \cdots, l + u$. It should be noted to keep the value of labeled nodes constant (0 or 1). The other nodes should have values that are calculated with the weighted average of their neighbors. The harmonic function is a real relaxation of the mincut function. Unlike mincut, which is not unique and is NP-hard, the minimum-energy harmonic function is unique and efficiently computable. Although harmonic function is not formulated as discrete values, one can use the domain knowledge or hard assignments to map from f to labels.

	Relevant	Not Relevant
Retrieved	true positive (tp)	false negative (fn)
Not Retrieved	false negative (fn)	true negative (tn)

Table 1: Retrieval contingency table

4 Evaluation in Information Retrieval

4.1 Overview

The first part of this study focuses on an overview of Information Retrieval (IR) evaluation in general. Concepts in this section are mainly referred to [49]. To evaluate the effectiveness of an IR system, one needs three things:

1. Document collection.
2. A set of queries.
3. A set of relevance judgment for each query. (Usually a binary judgment as *relevant* and *not relevant*)

In fact, a document is called relevant if it addresses the user's *information need.* The effectiveness of some IR systems depends on the value of a number of parameters. Manning et, al [49] adds that

> It is wrong to report results on a test collection which were obtained by tuning these parameters to maximize performance on that collection.

4.1.1 Binary Relevance, Set-based Evaluation

Two basic measures in IR system is precision and recall. *Precision* (P) is a fraction of retrieved documents that are relevant and *Recall* (R) is the fraction of relevant documents that are retrieved. According to table 1 the precision and recall values are calculated as:

$$P = tp/(tp + fp) \tag{1}$$
$$R = tp/(tp + fn) \tag{2}$$

Another measure to evaluate an IR system is *accuracy* which is the fraction of its classifications that are correct:

$$Accuracy = (tp + tn)/(tp + tn + fp + fn) \tag{3}$$

In most of the cases the data is extremely skewed: normally over 99.9% of the documents not relevant to the query [49]. Thus, trying to find some relevant documents will result in a large number of false positives.
A measure to make a trade off between precision and recall is the $F_measure$:

$$F = \frac{1}{\alpha/P + (1-p)/R} = \frac{(\beta^2+1)PR}{\beta^2 P + R}, \beta^2 = \frac{1-\alpha}{\alpha} \tag{4}$$

Values of $\beta < 1$ emphasize precision, while values of $\beta > 1$ have recall highly favored.

4.1.2 Evaluation of Ranked Retrieval

Measures mentioned in previous section are set-based measures, and use an unordered set of retrieved documents to evaluate the IR system. The first approach to evaluate a ranked retrieval is to draw the precision-recall plot. The precision-recall plot have a distinctive shape in which if the $(k+1)^{th}$ document retrieved is nonrelevant then recall is the same as for the top k documents, but precision has dropped. If it is relevant, then both precision and recall increase and the curve makes a shift to the right.

Usually the *interpolated precision (P_{interp})* is plotted. P_{interp} is the highest precision value of the system at a certain recall level. For a recall value r:

$$P_{interp}(r) = max_{r' \geqslant r} P(r') \tag{5}$$

The intuition behind this definition is that everybody would want to retrieve a few more documents if it would increase precision. The 11-point interpolated average precision is a traditional approach for this and was used in the first 8 TREC Ad Hoc evaluations. The interpolated precision is measured at the 11 recall levels between 0.0-1.0.

Another standard measure in TREC is Mean Average Precision (MAP) which reports a single value for the effectiveness of IR system. Average precision focuses on returning more relevant documents earlier. It is the average of precisions computed after cutting the list after each of the relevant documents retrieved in turn. The mean average precision is then the mean value of the average precisions computed for each of the queries separately. If the set of relevant documents for a query $q_j \in Q$ is $\{d_1, \cdots, d_{m_j}\}$ and R_{jk} is the set of ranked results from the top result until the document d_k then

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Precision(R_{jk}) \tag{6}$$

MAP is roughly the average area under the precision-recall curve for a set of queries.

Precision at k is a measure that finds the accuracy of an information retrieval system bashed on the first few pages of the results and does not require any estimate of the size of the set of relevant documents. An alternative for this measure is *R-Precision* which requires having a set of known relevant documents of size *R*. The precision is then calculated when the top *R* documents returned.

4.1.3 Non-binary Relevance Evaluation

Normalized discounted cumulative gain (NDCG) [38] is an increasingly adopted measure and is designed for situations of non-binary relevance and like precision at k, it is evaluated using k top search results.

If $R(j, d)$ is the relevance score the annotators gave to document d for query j then,

$$NDCG(Q, k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_k \sum_{m=1}^{k} \frac{2^{R(j,m)} - 1}{log(1+m)} \tag{7}$$

Z_k is a normalization factor calculated to ensure a perfect ranking's NDCG at k is 1.

		judge 2	Relevance	
		Yes	No	Total
Judge 1	Yes	300	20	320
Relevance	No	10	70	80
	Total	310	90	400

Table 2: Example on κ statistics

4.2 Assessing Agreement, Kappa Statistics

Once we have the set of documents and the set of queries we need to collect relevance assessments. This task is quite expensive in that judges should agree on the relevance of a document for each given query. The Kappa statistics [19] is

$$\kappa = \frac{\Pr(A) - \Pr(E)}{1 - \Pr(E)} \tag{8}$$

where $Pr(A)$ is the relative observed agreement among judges, and $Pr(E)$ is the probability that the judges agree by chance. The above formula is useful to calculate the kappa when there are only two judges. In the cases when there are more than two relevance judgments for each query Fleiss' kappa is calculated:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{9}$$

$1 - \bar{P}_e$ is the maximum agreement that is reachable above chance, and, $\bar{P} - \bar{P}_e$ gives the agreement actually which is actually achieved above chance. A κ value of 1 means complete agreement while it means disagreement when it is 0.

Krippendorff [41] discusses that finding associations between two variables that both rely on coding schemes with $\kappa < 0.7$ is often impossible, and that content analysis researchers generally think of $\kappa > 0.8$ as good reliability with $0.67 < \kappa < 0.8$ allowing provisional conclusions to be drawn. Kappa is widely accepted in the field of content analysis and is interpretable.

4.2.1 An Example

Table 2 shows an example from [49] on the judgments of two assessors.

$P(A) = (300 + 70)/400 = 370/400 = 0.925$
$P(nonrelevant) = (80 + 90)/(400 + 400) = 170/800 = 0.2125$
$P(relevant) = (320 + 310)/(400 + 400) = 630/800 = 0.7878$
$P(E) = P(nonrelevant)^2 + P(relevant)^2 = 0.2125^2 + 0.7878^2 = 0.665$
$\kappa = (P(A) - P(E))/(1 - P(E)) = (0.925 - 0.665)/(1 - 0.665) = 0.776$

4.3 Statistical testing of retrieval experiments

An evaluation study is not always complete without some measurement of the significance of the differences between retrieval methods. Hull [37] focuses on comparing two or more retrieval methods using statistical significance measures. The t-test compares the magnitude of difference between methods to the variation among the differences between the scores for each query which is analyzed. If the average difference is large comparing its standard error, then the methods are significantly different. The t-test assumes that the error follows a normal distribution. Two non parametric alternatives to t-test are paired Wilcoxon signed-rank test and the sign test. Let X_i and Y_i be the scores of retrieval methods X and Y for query i, and let $D_i = \theta + \epsilon_i$ where ϵ_i are independent. The null hypothesis is $\theta = 0$.

- **Paired t-test**

$$t = \frac{\bar{D}}{s(D_i)/\sqrt{n}}$$

- **Paired Wilcoxon test**

$$T = \frac{\sum R_i}{\sqrt{\sum R_i^2}}, R_i = sign(D_i) \times rank|D_i|$$

- **Sign Test**

$$T = \frac{2 \times \sum I[D_i < 0] - n}{\sqrt{n}}$$

where $I[D_i > 0] = 1$ if $D_i > 0$ and 0 otherwise.

5 Blog Analysis

5.1 Introduction

Nowadays *weblogs*, (aka. *blogs*) play an important role in social interactions. People who maintain blogs and update them, so called *bloggers*, involve in a series of interactions and interconnections with other people [12]. Each blog usually has a constant regular number of readers. These readers might make links to that blog or comment its posts which is said to be a motive for future postings [69].

Although, not until 1997, had the term *blog* been coined [2], today many people maintain blogs of their own and update it regularly, writing their feelings, thoughts, and any other thing they desire. The role of blogs as a social network is clear, and so, many research projects are devoted to analyze relations in blogs, from political conclusions [2] to finding mutual awareness and communities [45]. There are two reasons why blogs are systematically studied [42]:

1. **Sociological reasons**
   the cultural difference between the blogspace and regular webpages is that blogspace focuses heavily on local community interactions between a small number of bloggers. There is also a sequence of responces when hot topics arise. This makes it necessary to see if this dynamics can be explained and modeled.

2. **Technical reasons**
   The blogspace provides the notion of timestamp for blog posts and comments which makes it easier to analyze this space overtime. Analysis of the bursty evolution in blogs concentrates on bursts in the evolution of blogspace.

5.2 Implicit Structure of Blogs

Blogs are wonderful to analyze to track "memes" as they are constantly used and have and underlying structure. Blogs are significantly used to record diaries and are easy to update, so make the online document growth faster. In addition, the hyperlinks between bloggers form a network structure in the blogspace which makes the blogspace a wonderful testbench to analyze information dynamics in networks.

The microscopic patterns of blog epidemics are splitted up into implicit and explicit. Blog epidemics studies try to address two questions:

- Do different types of information have different epidemic profiles?
- Do different types of epidemic profiles group similar types of information?

General categories of information epidemics as well as introducing a tool to infer and visualize the paths specific infections in the blog network are the main contributions of [3]. Microscale dynamics of blogs is also studied in [3], where the authors specify two major factors to consider to address epidemics in blogs: Timing and graph. A few problems are to be considered in this analysis. First, The root is not always clearly identified. Second, there might be multiple possibilities for infection of a single node. As an example assume, A, B, C are blogs. B links to A and C links to both A, B. C might be infected either by B (After B is infected by A) or directly by A. Third, there is uncrawled space and the whole blogspace structure is not known to the analyst. Explicit

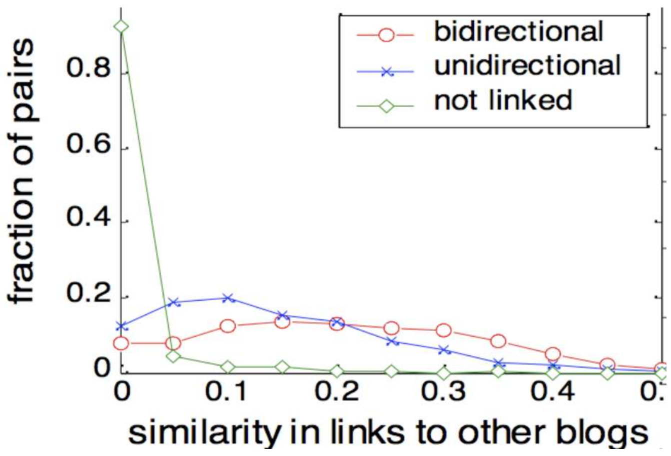

Figure 1: The results on the similarity of pairs in [3]

analysis is easy and can be done using the link structure. It is usually hard to analyze the implicit structure of blogs though. This is known as the link inference problem. Possible ways are to use machine learning algorithms such as support vector machines, or logistic regression. The full text of blogs, blogs in common, links in common, and history of infection are used as available source of information.

One experiment that Adar et, al [3] carry is to find pairs that are infected and are linked, and pairs that are infected but are unlinked. Although in general case the machine learning approach for both experiments shows more than 90 percent accuracy in the classification task, it doesn't work well in the specific case, in which for a given epidemic it connects all blogs.

Figure 1 is selected from [3] and shows the results on the distribution of similarity of pairs in links to other blogs.

5.2.1 iRank

Adar et, al [3] conclude with a description of a new ranking algorithm, iRank, for blogs. iRank, unlike traditional ranking algorithms, uses the implicit link structure to find those blogs that initiate the epidemics. It focuses on finding true information resources in a network. if n blogs are pointing to one, and that one is pointing to another single information resource the latter might be the true source of information. In the iRank algorithm first, a weighted edge is drawn for every pair of blogs that cite the same url, u:

$$W_{ij}^{u} = W(\Delta d_{ij})$$

Weights are then normalized so outgoing weights sum to one, and at last a pagerank is performed on the weighted graph.

5.3 Information Diffusion in Blogs

Dynamics of information in the blogosphere is studied in [36] and is the main focus of this section. Over the past two decades there has been an increasing interest in observing information flows as well as influencing and creating theses flows. The information diffusion is characterized along two dimensions:

- **Topics**
  This includes identifying a set of postings that are about the same topic, and then characterizing the different patterns the set of postings may fall in. Topics, according to [36] are unions of chatter and spikes. The former refers to the discussions whose subtopic flow is determined by authors' decisions. The latter refers to short-term, high intensity discussions of real world events.

- **Individuals**
  The individual behavior in publishing online diaries differs dramatically. [36] characterize the four categories of individuals based on their typical posting behavior within life cycle of a topic.

The corpus was collected by crawling the web of 11,804 RSS blog feeds. Gruhl et, al [36] collected 2K-10K blog postings per day for a total of 401,021 postings in their data set. Based on their observations on manually analyzing 12K individual words highly ranked using tf-idf method, they decompose topics along two axes: chatters (internally driven and sustained discussions) and spikes (externally induced sharp rises in postings). Depending on the average chatter level topics can be places into one of the following three categories:

1. Just Spike

2. Spiky Chatter (Topics that are significantly chatter, but alongside, have high sensitivity to external events)

3. Mostly Chatter

The model of "Topic = Chatter + Spikes" is further refined to see if Spikes themselves are decomposable. For each proper noun x they compute the support s (the number of times x co-occurred with the target) and the reverse confidence $c_r = P(target|x)$. To generate a rational term set, thresholds for s and c_r are used. For these terms they look at the co-occurrences and define spikes as areas where the posts in a given day exceed a certain number.

The distribution of non-spike daily average is approximated by

$$Pr(\text{avg \# of posts per day} > x) \approx ce^{-x}$$

This observation is also made that most spikes in the manually labeled chatter topics last about 510 days.

Table 3 shows the distribution of number of posts by user. The authors in [36] try to locate users whose posts fall in the categories show in table 3. Each post on day i has a probability p_i that falls into a given category.

They gather all blog posts that contain a particular topic into a list

$$[(u_1, t_1), (u_2, t_2), \cdots, (u_k, t_k)]$$

sorted by publication date of the blog, in which u_i is the ID of the blog i, and t_i is the first time at which blog u_i contained the reference to the topic. The aim is then to induce the relevant edges among a candidate set of $\theta(n^2)$ edges.

Predicate	Algorithm	Region	% of Topics
RampUp	All days in the first 20% of post mass below mean and average day during this period below $\mu - \sigma/2$	first 20% of posts mass	3.7%
Ramp-Down	All days in the last 20% of post mass below mean and average day during this period below $\mu - \sigma/2$	last 20% of posts mass	5.1%
Mid-High	All days during middle 25% of post mass above mean and average day during this period below $\mu - \sigma/2$	Middle 25% of posts mass	9.4%
Spike	For some day, number of posts exceed $\mu - \sigma/2$	From Spike to infection point below μ both directions.	18.2%

Table 3: Distribution of the number of posts by user

If a appears in the traversal sequence and b does not appear later in the same sequence, this shows valuable information about (a, b). If b were a regular reader of a then, memes discussed by a should sometimes appear in b.

The authors present an EM like algorithm to induce the parameter of the transmission graph, in which they first compute soft assignments of each new edge infection, and then update the edge parameters to increase the likelihood of assigned infections. They estimate the copy probability κ, and inverse mean propagation delay (r) as:

$$r = \frac{\sum_{j \in S_1} p_j}{\sum_{j \in S_1} p_j \delta_j}$$

and

$$\kappa = \frac{\sum_{j \in S_1} p_j}{\sum_{j \in S_1 \cup S_2} Pr(r \leqslant \delta_j)}$$

where $Pr(a < b) = (1-a)(1-(a-1)^b)$ the probability that a geometric distribution with parameter a has value $\leqslant b$.

5.4 Bursty Evolution

The study of evolution of blogs is tightly coupled with the notion of a *timed graph*. There is a clear difference between the community structures in the web and that of blogs. Blog communities are usually formed as a result of a debate over time which leaves a number of hyperlinks in the blogspace. Therefore, the community structure in blogs should be studied during short intervals, as the heavy linkage in a short period of time is not significant when analyzed over a long time span.

5.4.1 Time Graphs

A time graph $G = (V, E)$ consists of

1. A set V of nodes, with each node associated with a time interval $D(v)$ (duration).

2. A set E of edges where each edge is defined as a triple: $e = (u, v, t)$ where u and v are nodes and t is a point in the time interval $D(u) \cap D(v)$

5.4.2 Method

The algorithm in [42] consists of two steps: pruning, and expansion. The pruning step is simply done by first removing nodes with degree zero and one, and then checking all nodes with degree 2, to see if their neighbors are connected and form triangles. In that case, they are passed to the expansion procedure (i.e. growing the seed to a dense subgraph to find a dense community.) and the resulted expansion is reported as a community and is removed from the graph.

The results of studying the SCC evolution in blogs show that for each of the three largest strongly connected components, at the beginning of the study, the number of blog pages is significant but there is no strongly connected component of more than a few nodes. Around the beginning of the second year, a few components representing 1-2% of the nodes in the graph appear, and maintain the same relative size for the next year. In the forth year, however, the size of the component increases to over 20% by the present day. The giant component still appears to be expanding rapidly, doubling in size approximately every three months.

Results in [42] also indicate that the number of nodes in the communities for randomized blogspace is an order of magnitude smaller than for blogspace. This shows that the community formation in blogspace is not a property of the growth of the graph. In addition, the SSC in randomized blogspace grows much faster than in the original blogspace. This means that the striking community in the blogspace is a result of the links which are in deed referenced to topicality.

6 Lexical Networks

6.1 Introduction

One of the major differences between humans and other species is that human beings are capable of adapting languages that are not shared by any other species. Human language allows the construction of a virtually infinite range of combinations from a limited set of basic units. We are able to rapidly produce words to form sentences in a highly reliable fashion as the process of sentence generation is very rapid. Generation and evolution of languages causes the creation of new language entities.

In this chapter we look at the some previous work on lexical networks. A *lexical network* refers to a complete weighted graph in which vertices represent linguistic entities, such as words or documents, and edge weights show a relationship between two nodes. Three main classes of lexical networks are word-based, sentence-based, and document-based networks. Word-based lexical networks can be further divided into networks based on co-occurrence, syllabus, semantic networks, and synthetic networks.

6.1.1 Small World of Human Languages

The small world property [72] of language networks has been analyzed before [67]. Ferrer Cancho et, al [32] showed that the syntactic network of heads and their modifiers exhibits small-world properties. The small-world property is also shown for the co-occurrence networks by [31]. A co-occurrence network is a network of words as nodes and an edge appear between two nodes if they appear in the same sentence. Many co-occurrences are due to syntactical relationships between words or dependency relationships [53].
In this section we take a deeper look at the work by [31]. They set the maximum distance according to assume a co-occurrence to be the minimum distance at which most of the co-occurrences are likely to happen:

- Many co-occurrences take place at a distance of one.
- Many co-occurrences take place at a distance of two.

It is also shown before that co-occurrences happen at distances greater than two [22]. However, for four reasons they decide to just consider maximum distance of two for a co-occurrence:

1. Unavailability of a system to perform the task of considering any distances greater than two, and that all achievements in computational linguistics are based on consideration of a maximum distance of two.
2. Method failure in capturing exact relationships at distances greater than two.
3. To make this as automatic as possible they stick with distance two and try to collect as many links as possible.
4. long distance syntactic dependencies imply smaller syntactic links.

The authors consider the graph human language, Ω_L, as defined by $\Omega_L(W_L, E_L)$, where $W_L = \{w_i\}$, $(i = 1, \cdots, N_L)$ is the set of N_L words and $E_L = \{w_i, w_j\}$ is the set of edges or connections between words. They call the biggest connected component of the networks that results from the basic and improved methods,, respectively, the unrestricted word network (UWN) and the restricted word network (RWN). They analyze the degree distribution for the unrestricted word network and the restricted word network of about three quarters of the 107 words of the British National Corpus (http://info.ox.ac.uk/bnc/). The authors further show that the degree increases as a function of frequency, with exponent 0.80 for the first and 0.66 for the second segment.

In summary, they show that the graph connecting words in language exhibits the same statistical features as other complex networks. The observe short distances between words arising from the small-world structure. They conclude that language evolution might have involved the selection of a particular arrangement of connections between words.

6.2 Language Growth Model

In this section we summarized the work by Dorogovtsev and Mendes [27] which describes the evolution of language networks. Human language can be described as a network of linked words. In this network neighboring words in language sentences are connected by edges. Dorogovtsev and Mendes [27] treat the language as a self-organizing network and in which nodes interact with each other and the network grows. They describe a network growth with the following rules:

- At each time step, a new vertex (word) is added to the network.
- A new node at its birth, connects to several old nodes whose number is of the order of 1.
- For convention it is assumed that a new node only attaches to one node with the probability proportional to its degree.
- At each time increment, t, ct edges emerge in the network between old words where c is constant.
- New edges connect nodes with probability proportional to their degree.

In continues approximation, the degree of a vertex born at i and observed at t is described by

$$\frac{\partial k(i,t)}{\partial t} = (1+2ct)\frac{k(i,t)}{\int_0^t k(u,t)du}$$

The total degree of the network at time t is

$$\int_0^t k(u,t)du = 2t + ct^2$$

Solving for $k(i,t)$ with the initial condition, $k(t,t) = 1$ results:

$$k(i,t) = \Big(\frac{ct}{ci}\Big)^{1/2}\Big(\frac{2+ct}{2+ci}\Big)^{3/2}$$

The nonstationary degree distribution is then

$$P(k,t) = \frac{1}{ct}\frac{ci(2+ci)}{1+2ci}\frac{1}{k}$$

This results in a power-law degree distribution with two different regions, intersecting at

$$k_{cross} \approx \sqrt{ct}(2+ct)^{3/2}$$

Below this point, the degree distribution is stationary:

$$P(k) \cong 1/2k^{-3/2}$$

And above the cross point we they obtain:

$$P(k,t) \cong 1/4(ct)^3k^-3$$

which is a nonstationary degree distribution in this region

6.3 Complex Networks and Text Quality

6.3.1 Text Quality

The first work at which we take a closer look is the text assessment method described by Antiqueria et, al [6] who looked at the text assessment problem using the concept of complex networks. In their work they model the text with a complex network, in which each of the N words is represented as a node and each connection between words as a weighted edge between the respective nodes representing those words. They also get use of a list of stop words to eliminate terms not associated with concepts.

The define two measures, *instrength* and *outstrength* corresponding to weighted indegree and outdegree respectively as

$$K_{in}(i) = \sum_{j=1}^{N} W(j,i)$$

and

$$K_{out}(i) = \sum_{j=1}^{N} W(i,j)$$

Antiqueria et, al [6] reach several conclusions:

1. The quality of the text tends to decrease with outdegree.
2. Quality is almost independent (increases only slightly) of the number of outdegrees for the good-quality texts, while for the low-quality texts the quality increases with the number of outdegrees.
3. The quality of the text decreases with the clustering coefficient.
4. In low-quality texts there is much higher variability in the clustering coefficients, which also tend to be high.
5. Quality tends to increase with the size of the minimum path for all the three definitions of path used in their work.

6.3.2 Summary Evaluation

A summary evaluation method based on complex networks in discussed in [62]. The first step to define a measure to evaluate summaries using complex network concepts is to represent summaries as complex networks. In this representation terms of the summary are nodes in the network, and each association is determined by a simple adjacency relation. There is an edge between every adjacent words in the summary. The weight on edges show the number of times the two terms are adjacent in the summary.

By modeling summaries as complex networks and by introducing a network metric, the authors showed it possible to evaluate different automatic summaries. Their measure is based on the number of strongly connected components in the network. More formally, they define a deviation measure:

$$deviation = \frac{\sum_{M=1}^{A} |f(M) - g(M)|/N}{A}$$

where $f(M)$ is the function that determines the number of components for M words associations and $g(M)$ is the function that determines the linear variation of components for M words associations. In this measure N is the number of different words in the text and A is the total number of words associations found. Their measure is evaluated by using it to evaluate three automatic summarizers: GistSumm [61], GEI [60], and SuPor [56] against a manual summary.

6.4 Compositionality in Quantitative Semantics

The last article that we review in this chapter is on compositionality in quantitative semantics by [52]. They introduce a principle of latent compositionality into quantitative semantics. They implement a Hierarchical Constraint Satisfaction Problem (HCSP) as the fundamental text representation model. This implementation utilizes an extended model of semantic spaces which is sensitive to text structure and thus leaves behind the bag-of-feature approach. A simple example shows the different models' behavior in interpreting a sentence

All sculptures are restored. Only the lions stay in the garden.

- **Vector Space (VS)**
The representation of this text in the vector space model is a bag of words in which stop words are filtered out. Important words (garden, lion, restore, sculpture, stay) are used to build a vector of weighted terms to represent the sentence in the model.

- **Latent Semantic Analysis (LSA)**
LSA extends the previous approach. Unlike the VS model, it does not refer to a weighted bag of words, but utilizes factor analytic representations. In this model, the sample is represented by the strongest factors in locating the representations of the input words garden, lion, restore, sculpture, stay within the semantic space. Therefore, this can be assumed of a bag of feature-vector representation but it still ignores the structure of the text.

- **Fuzzy Linguistics (FL)**
FL derives a representation of text based on its integration hierarchy. This model expects the text as a whole suggest words like sculptural art, park, collection, but not veldt, elephant, or zoo. This analysis assumes the present order of sentences. It does not allow reinterpreting

lion if the order of the sentences is reversed. Moreover, the example presupposes that all coherence relations as well as the integration hierarchy have been identified before.

The major contribution of [52] is, however, a formalization of the principle of compositionality in terms of semantic spaces. In particular, they try to fill the gap of the missing linguistic interpretability of statistical meaning representations.

7 Text Summarization

This chapter covers some of the previous salient works on text summarization.

7.1 LexRank

Lexrank is an stochastic graph based method to determine the lexical centrality of documents within a cluster, and is introduced in [30]. The network on which lexrank is applied is a complete weighted network of linguistic entities. These entities can be sentences, or documents. In its special application for summarization, lexrank is used to find the centrality of sentences within a text document.

To form the network, Lexrank utilizes the tf-idf term vector and the so called "bag of words" representation. This enables Lexrank to use the cosine similarity of documents as weight edges to build the entire network. The cosine similarity of two document vectors is calculated as

$$Sim(\overrightarrow{d_i}, \overrightarrow{d_j}) = \frac{\overrightarrow{d_i} \cdot \overrightarrow{d_j}}{|d_i||d_j|}$$

In this representation of documents, $\overrightarrow{d_i} = (w_{1i}, w_{2i}, \cdots, w_{Vi})$, where V is the size of the vocabulary and

$$w_{ji} = tf_{ji} \cdot idf_j$$

tf_{ji} is the *normalized term frequency* of term j in document i and idf_j is the *inverse document frequency* of the term j.

The centrality of a node u in this network is related to its neighbors' centralities and is calculated as:

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{Sim(u,v)}{\sum_{z \in adj[v]} Sim(z,v)} p(v)$$

d in this equation is the damping factor to ensure convergence of the values.

Lexrank is evaluated using 30 documents of DUC 2003 and 50 documents of 2004 using ROUGE[1] evaluation system.

7.2 Summarization of Medical Documents

In this section we review the work in [4] whose main aim is to survey the recent work in medical documents summarization. One of the major problems with the the medical domain is that it suffers particularly from information overload. This is important to deal with as it is necessary for physicians to be able to quickly and efficiently access to up-to-date information according to their particular needs. The need for an automatic summarization system is more crucial in this field as a result of number and diversity of medical information sources. These summarizers help medical researchers determine the main points of a document as quickly as possible.

A number of factors have to be taken into consideration while developing a summarization system:

[1]http://www.isi.edu/ cyl/ROUGE

- **Input**
  One major factor is the type of the input to such system. Input can be *Single Document* or *Multi Document.* Additionally input *language* should be taken into account, which results in three different type of summarizing systems: *monolingual*, *multilingual*, or *cross-lingual.* The third input factor is type of the input, which can be anything from plain text to images and sound.

- **Purpose**
  These factors concern the possible uses of the summary, and the potential readers of the summary. A summary can be *indicative* or *informative.* The former does not claim any role of substituting the source documents, while the latter substitutes the original documents. In another categorization of summaries, a summary can be *generic* versus *user-oriented.* Generic systems create a summary of a document or a set of documents based on all the information found in the documents. User-oriented systems create outputs that are relevant to a given input query. From another perspective a summary can either be *general purpose* or *domain-specific.*

- **Output**
  Evaluation type of an automatic summarizing system determines its type of output. It can be either *qualitative* or it can be *quantitative.* The last, yet major factor in creating a summary is considering the relation that the summary has to the source document. A summary can be an abstract, or an extract. Extracts include material (i.e., sentences, paragraphs, or even phrases) from the source documents. An abstract, on the other hand, tries to identify the most salient concepts in the source documents,then utilizes natural language processing techniques to neatly present them. Generation.

7.3 Summarization Evaluation

The existing evaluation metrics can be split into two categories: *intrinsic* and *extrinsic.* An intrinsic method evaluates the system summary independently of the purpose that the summary is supposed to satisfy. An extrinsic evaluation, on the other hand, evaluates the produced summary in terms of a specific task. The quality measures which are used by the intrinsic methos can be the integrity of its sentences, the existence of anaphors, the summary readability, the fidelity of the summary compared to the source documents. Gold summary is another way of doing an intrinsic evaluation. Gold summaries are human-made summaries that are ideal and can be used to compare to system summaries. However, it is usually hard to make annotators agree on what constitutes a “gold” summary.

Tables 4, 5 summarize some of the extractive and abstractive methods of text summarization respectively.

7.4 MMR

In this section we reviewed the MMR method which is a re-ranking algorithm based on diversity which is used to produce summaries. Conventional IR methods maximize the relevance of the retrieved documents to the query. This is, however, tricky when there are many potentially relevant documents with partial information overlap. Maximal Marginal Relevance (MMR) is a user-tunable

Input	Purpose	Output	Method	Evaluation	ref.
Single-document, English, text	Generic, domain-specific (technical papers)	Sentences	Statistics (Edmundsonian, paradigm) no revision		[46]
Single-document, English, text	Generic, domain-specific (scientific articles on specific topics)	Sentences	Use of thematic keywords, no revision, Statistics (Edmundsonian paradigm)	Intrinsic	[29]
Single-document, multilingual, text	Generic, domain-specific (news)	Sentences	Statistics (Edmundsonian paradigm), no revision	Intrinsic	[24]
Single-document, multilingual, text	User-oriented, domain-specific (scientific and technical texts)	Sentences	Statistics (Edmundsonian paradigm), use of thesauri, revision		[1]
Multi-document, multilingual , text (English, Chinese)	Generic, domain-specific (news)	Sentences	Language processing (to identify keywords)	Extrinsic	[20]
Single-document, English, text	Generic, general purpose	Paragraphs	Graph-based, statistics (cosine similarity, vector space model)	Intrinsic	[66]
Multi-document, English, text	User-oriented, General purpose	Text regions (sentences, paragraphs, sections)	Graph-based, cohesion relations, language processing	Intrinsic, extrinsic	[48]
Single-document, English, text	Generic, domain-specific (scientific articles)	Sentences	Tree-based, language processing (to identify the) (RST relations markers)	Intrinsic, extrinsic	[50, 51]

Table 4: Summarizing systems with extractive techniques

Input	Purpose	Output	Method	Evaluation	ref.
Single-document, English, text	Informative, user-oriented, domain-specific	Scripts	Script activation, canned generation		[25]
Multi-document, English, text	Informative, user-oriented, domain-specific	Templates	Information extraction, NLG	Evaluation of System components	[64]
Single-document, English, text	Generic, domain-specific (news articles)	Clusters	Syntactic processing of Representative Sentences, NLG	Intrinsic	[10]
Single-document, English, text	Informative, user-oriented, domain-specific	Ontology-based representation	Syntactic processing of Representative Sentences, ontology-based annotation, NLG	Extrinsic	[65]
Single-document, multilingual, text		Conceptual representation in UNL	Statistics (for scoring each UNL sentence), removing redundant words, combining sentences		[27]

Table 5: Summarizing systems with abstractive techniques

method and a re-ranking method with a functionality to drill down on a narrow topic or retrieving a large range of relevance bearing documents.

The criterion considered in MMR is the relevant novelty. The first way to do this is to measure novelty and relevance independently, and then using a linear combination of two metrics provide a novelty-relevance measure. This linear combination is called the *Marginal Relevance*. So a document will have a high marginal relevance if it is both relevant and has the minimum information overlap with the previously selected documents.

$$MMR = argmax_{D_i \in R \setminus S}[\lambda(Sim_1(D_i, Q) - (1-\lambda)max_{D_j \in S} Sim_2(D_i, D_j))]$$

with Q begin the query. This can be specifically used in summarization. In that case, the top highly ranked passages of a document can be chosen to be included in the summary. This summarization is shown to work better for longer documents (which contain more passage redundancy across document sections [18]. MMR is also extremely useful in extracting passages from multiple documents that are about the same topics.

8 Graph Mincuts

This chapter reviews some of the previous salient works on graph mincut utilization for solving machine learning related problems.

8.1 Semi-supervised Learning

The major issue in all learning algorithms is the lack of sufficient labeled data. Learning methods are usually used in classifying text, web pages, and images, and they need sufficient annotated corpora. Unlike labeled data, whose creation is quite tedious, unlabeled data is quite easy to gather. For example, in a classification problem, one can easily access a large number of unlabeled text documents, but the number of manually labeled data can hardly exceed a few. This causes a great interest in semi-supervised methods.

The first paper that we review in this work uses graph mincuts to classify data points using both labeled and unlabeled data. This work utilizes the idea of graph mincuts [15]. Given a combination of labeled and unlabeled datasets, Blum and Chawla [15] construct a graph of the examples such that the minimum cut on the graph yields an "optimal" binary labeling data according to some predefined optimization function.

In particular, the goal in [15] is to

1. Find the global optimum which is better than a local optimum regarding the objective minimization function.

2. Utilize this method to reach a significant difference in terms of prediction accuracy.

Blum and Chawla [15] also make an assumption that the unlabeled data comes under the same distribution that the labeled data does. The classification algorithm described in [15] has five steps:

1. Construct a weighted graph $G = (V, E)$ where V is the set of all data points as well as a sink v_- and a source v_+. The sink and the source are classification nodes.

2. Connect classification nodes to those *labeled* examples that have the same label with edges having infinite weights. That is, for all $v \in +$ add $e(v_+, v) = \infty$ and for all $v \in -$, add $e(v_+, v) = \infty$.

3. Weights between example vertices are assigned using a similarity/distance function.

4. Determine the min-cut of the graph. This means to find a minimum total weight set of edges whose removal discounts v_+ from v_-. The removal of edges on the cut will make a graph be split into two parts: V_+ and V_-.

5. Assign positive labels to all nodes in V_+ and negative labels to all nodes in V_-.

8.1.1 Why This Works

The correctness of this algorithm is dependent on the choice of the weight function. For a certain learning algorithm, we can define edge weights so that the mincut algorithm produces a labeling which minimizes the leave one out cross validation error when applied to entire $U \cup L$. Additionally, for certain learning algorithms, we can also define the edge weights so that the mincut algorithm's labeling results in a zero leave one out cross validation error in L. This is correct according to the following discussions: (See [15] for proof)

> If we define edge weights between examples in a way that for each pair of nodes x, y, we have $nn_{xy} = 1$ if y is the nearest neighbor of x, and $nn_{xy} = 0$ otherwise, and also define $w(x, y) = nn_{xy} + nn_{yx}$, then for any binary labeling of x, the cost of associated cut is equal to the number of leave-one-out cross validation mistakes made by 1-nearest neighbor on $L \cup U$.
>
> Given a locally weighted averaging algorithm, we can define edge weights so that the minimum cut yields a labeling that minimizes the L_1-norm leave one out cross validation error.
>
> Let w be the weight function used for the symmetric weighted nearest neighbor algorithm, then if the same function is used for finding the graph mincut, the classification resulted by this method has a zero leave one out cross validation error in U.
>
> Suppose the data is generated at random in the set of k $(\epsilon, \delta/4)$-round regions, such that the distance between any two regions is at least δ an the classification only depends on the region to which a point belongs to. If the weighting function is $w(x, y) = 1$ if $d(x, y) < \delta$ and $w(x, y) = 0$ otherwise, then $O((k/\epsilon) \log k$ labeled examples and $O((1/V_{\delta/4}) \log(1/V_{\delta/8})$ unlabeled examples are sufficient to classify a $1 - O(\epsilon)$ fraction of an unlabeled examples correctly.

Blum and Chawla [15] show the efficiency of their method, using standard data as well as synthetic corpora they also show that this method is robust to noise.

8.2 Randomized Mincuts

The method of randomized mincuts extends the mincut approach by adding some sort of randomness to the graph structure, with some preliminary assumptions. Blum et, al [16] assume that similar examples have similar labels. So a natural approach to use unlabeled data is to combine nearest-neighbor prediction. In other words, as an example, similar unlabeled data should be put into same classes.

The mincut approach for classification has several properties. First, it can be easily found in polynomial time using network flow. Second, it can be viewed as giving the most probable configuration of labels in the associated Markov random field. However, this method has some shortcomings. Consider, as an example, a line of n vertices between two points s, t. This graph has $n - 1$ cuts of size 1 and the cut at the very left will be quite unbalanced.
The method described in [16] proposes a simple method or addressing a number of these drawbacks using a randomization approach. Specifically, the method repeatedly adds random noise to the edge weights. Then it solves the mincut for the graph and outputs a fractional label for each example. In this section we take a closer look at this method described in [16].

A natural energy function to consider is

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij} |f(i) - f(j)| = \frac{1}{4} \sum_{i,j} w_{ij} (f(i) - (f(j))^2$$

where the partition function Z normalizes over all labeling. Solving for the lowest energy configuration in this Markov random field produces a partition of the whole dataset and maximizes self-consistency. The randomized mincut procedure is as follows:
Given a graph G constructed from the dataset, produce a collection of cuts by repeatedly adding random noise to the edge weights and then solve the mincut in the resulted graph. Finally remove cuts which are highly unbalanced. Blum et, al [16] show that we only need $O(k \log n)$ labeled examples to be confident in a consistent of k edges.

8.2.1 Graph Construction

For a given distance metric, we can construct a graph in various ways. The graph should be either connected or a small number of connected components cover all examples. If t components are needed to cover a $1 - \epsilon$ fraction of the points, the graph based method will then need at least t examples to perform well. Blum et, al [16] construct the graph by simply making a minimum spanning tree on the entire dataset. Their experimental setup is to analyze three datasets: handwritten digits, newsgroups, and the UCI dataset. The results from these experiments suggest the applicability of the randomized mincut algorithm to various settings.

8.3 Sentiment Analysis

The analysis of different opinions and subjectivity has received a great amount of attention lately because of its various applications. Pang and Lee [59] discuss a new approach in sentiment analysis using graph mincuts. Previous approaches focused on selecting lexical features, and classified sentences based on some such features. Conversely, the method in [59] (1) labels the sentences in the document as either subjective or objective and then (2) applies a standard machine learning classifier to make an extract.

Document polarity can be considered as a special case of text classification with sentiment rather than topic based categories. Therefore, one can apply standard machine learning techniques to this problem. The pipeline in extracting the sentiments is like the following process:

n-sentence Review → subjectivity tagging → m-sentence extraction → positive/negative

The cut based classification utilizes two weight functions: $ind_j(x)$, and $assoc(x, y)$. The definition of these functions will be discussed later in this chapter. Here, $ind_j(x)$ shows the closeness of data point x to class j and the association function shows the similarity of two data points to each other.

The mincut minimizes

$$\sum_{x \in C_1} ind_2(x) + \sum_{x \in C_2} ind_2(x) + \sum_{x_i \in C_1, x_k \in C_2} assoc(x_i, x_k)$$

It should be noted that every cut corresponds to a partition of items that has a cost equal to the partition cost, and the mincut minimizes this cost. The set individual scores $ind_1(x)$ to $Pr^{NB}_{sub}(x)$, and $ind_2(x)$ to $1 - Pr^{NB}_{sub}(x)$, where $Pr^{NB}_{sub}(x)$ is Naive Bayes' estimate of the probability that sentence x is subjective (Weights of SVM can also be used). The degree of proximity is used as the association score of two nodes.

$$assoc(x_i, x_j) = \begin{cases} f(j-i) \cdot c & \text{if } j - i \leqslant T \\ 0 & \text{otherwise} \end{cases}$$

$f(d)$ specifies how the influence of proximal sentences decays with respect to distance d. Pang and Lee [59] use $f(d) = 1$, e^{1-d}, and $1/d^2$. They show that the for Naive Bayes polarity classifier, the subjectivity extracts are shown to be more effective input than the original documents. They also conclude that employing the minimum cut framework may result in an efficient algorithm for sentiment analysis.

8.4 Energy Minimization

The last work that we took a look at is a method of fast approximation for energy minimization using mincuts, described in [17].
The motivation in this work comes from Computer Vision. As mentioned before, these problems can be naturally formulated in terms of energy minimization. This formulation means that one aims to find a labeling function f that minimizes the energy

$$E(f) = E_{smooth}(f) + E_{data}(f)$$

in which E_{smooth} is a functions that measures the extent to which f is not smooth, and E_{data} shows the disagreement between f and observed data. Here E_{data} can be

$$E_{data}(f) = \sum_{pinP} (f_p - I_p)^2$$

with I_p being the observed intensity of p. The main problems in dealing with minimization problems is the large computational costs. Normally, these minimization functions have several local minima and finding the global minimum is a difficult problem, as the space of possible labeling has a big dimension, $|P|$ and can be up to many thousands. The energy function can also be considered as

$$E(f) = \sum_{\{p,q\} \in N} V_{p,q}(f_p, f_q) + \sum_{p \in P} D_p(f_p)$$

where N is set of interacting pairs of pixels (typically, adjacent pixels).

The method described in [17] generates a local minimum with respect to two types of very large moves $\alpha - expansion$ and $\alpha - \beta - swap$.

8.4.1 Moves

A labeling f can be represented by a partition of image pixels $P = \{P_l | l \in L\}$, and $P_l = \{p \in P | f_p = l\}$ is the subset of pixels labeled l. Given a pair of labels α, β, a move from partition P to a new partition P' is called a $\alpha - \beta - swap$ if $P_l = P'_l$ for any label $l \neq \alpha, \beta$. The $\alpha - \beta - swap$ means that the only difference between P and P' is that some pixels labeled α in P are now labeled β in P' and some pixels labeled β in P are now labeled α in P'.

A move from partition P to a new partition P' is called $\alpha - expansion$ for a label α if $P_\alpha \subset P'_\alpha$ and $P'_l \subset P_l$ for any label $l \neq \alpha$. This moves allows a set of image pixels to change their labels to α.

Two moves are described in the following:

```
$\alpha - \beta - swap$:
1. Start with an arbitrary labeling $f$.
2. Set $Success = 0$.
3. For each pair of labels $\{\alpha, \beta\} \in L$
       Find $\hat{f} = \arg\min E(f')$ among $f'$ within one $\alpha - \beta - swap$.
       If $E(\hat{f}) < E(f)$, set $f = \hat{f}$ and $Success = 1$
4. If $Success = 1$ goto 2.
5. Return $f$
```

```
$expansion$
1. Start with an arbitrary labeling $f$.
2. Set $Success = 0$.
3. For each label $\alpha \in L$
       Find $\hat{f} = \arg\min E(f')$ among $f'$ within one $\alpha - expansion$ of $f$.
       If $E(\hat{f}) < E(f)$, set $f = \hat{f}$ and $Success = 1$
4. If $Success = 1$ goto 2.
5. Return $f$
```

Now given an input labeling f and a pair of labels, α, β they find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta - swap$ of f. Also, they find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - expansion$ of f given an input labeling f, and a label α. They do this part using graph mincuts. They further prove that, a local minimum $\hat{f}$, when expansion moves are allowed, and $f*$ as the global minimum, satisfies

$$E(\hat{f}) \leqslant 2cE(f*)$$

To evaluate their method, they present the results of experiments on visual correspondence for stereo, motion, and image restoration.. In image restoration they try to restore the original image from a noisy and affected image. In this example, labels are all possible colors.

9 Graph Learning I

In this chapter and the next, we review some prior learning techniques which use the graph setting as the framework. We look at an application on text summarization, the problem of best outbreak in blog networks and those of sensor placement, the problem of web projection, and the co-clustering technique.

9.1 Summarization

The first section of this chapter describes a summarization method based on the approach in [75]. Zha [75] describes text summarization whose goal is to take a textual document, extract appropriate content, and present the most important facts of the text to the user, which matches the users' need.

Zha [75] adopts an unsupervised approach, and explicitly model both keyphrase and sentences that contain them, using the concept of bipartite graphs. For each document, they generate two sets of objects: the set of terms $T = \{t_1, t_2, \cdots, t_n\}$ and the set of sentences $S = \{s_1, s_2, \cdots, s_m\}$. A bipartite graph is then constructed using these two sets from T to S, in a way that there is an edge between t_i and s_j if t_i appears in s_j. The weight of the edge is a nonnegative value, and can be set proportional to the number of times t_i appears in s_j.

The major principle in this paper is the following:

> A term should have a high salience score if it appears in many sentences with high salience scores, while a sentence should have a high salience score if it contains many terms with high salience scores.

More formally,

$$u(t_i) \propto \sum_{v(s_j) \sim u(t_i)} w_{ij} v(s_j)$$
$$v(s_j) \propto \sum_{u(t_i) \sim v(s_j)} w_{ij} u(t_i)$$

where $a \sim b$ shows the existence of an edge between a, b. The matrix form of these equations will be

$$u = \frac{1}{\sigma} W v$$
$$v = \frac{1}{\sigma} W^T u$$

It is then clear that u, v are the left and right singular vectors of W corresponding the singular value σ and that, if σ is the largest singular value of W, then both u and v have nonnegative components. For numerical computation of the largest singular value triple $\{u, \sigma, v\}$, Zha [75] uses a variation of the power method. Specifically, the author chooses an initial value for v to be the vector of all ones. The following equations are then performed until convergence is achieved.

1. $u = Wv, u = u/\|u\|$

2. $v = W^T u, v = v/\|v\|$

Zha [75] uses this salience score to identify salient sentences within topics. For this purpose they use clustering. For sentence clustering, they build an undirected weighted graph with sentences as nodes, and edges representing the fact that two nodes (sentences) are sharing a term. The weight w_{ij} is considered to be as sparse as $W^T W$.

Two sentences s_i and s_j are said to be *near-by* if one follows the other in the linear order of the document. Then, set

$$\hat{w}_{ij} = \begin{cases} w_{ij} + \alpha & \text{if } s_i \text{ and } s_j \text{ are nearby} \\ w_{ij} & \text{otherwise} \end{cases}$$

For a fixed α they then apply the spectral clustering technique to obtain a set of sentence clusters $\Pi^*(\alpha)$. Thus, the problem is reduced to minimizing a clustering cost function defined as

$$GCV(\alpha) = (n - k - J(W, \Pi^*(\alpha)))/\gamma(\Pi^*(\alpha))$$

where k is the number of desired sentence clusters, W is the weight matrix for term-sentence bipartite graph and $\Pi^*(\alpha)$ is the set of clusters obtained by applying the spectral clustering to the modified weight matrix $W(\alpha)$. Also

$$J(W, \Pi) = \sum_{i=1}^{k} \sum_{s=1}^{n_i} ||w_s^{(i)} - m_i||^2$$

and,

$$m_i = \sum_{s=1}^{n_i} w_s^{(i)}/n_i$$

They then use the traditional K-means algorithm iteratively and in each iteration do the following steps:

1. For each sentence vector w, find the center m_i that is closest to w, and associate w with this cluster
2. Compute the new set of centers by taking the center of mass of sentence vectors associated with that center.

This way they find the local minimum of $J(W, \Pi)$ with respect to Π. They then use sum-of-squares formulation as a matrix trace maximization with special constraints, relaxing which, leads to a trace maximization problem that possesses optimal global solution. Formally, let

$$m_i = W_i e/n_i$$

and

$$X = diag(e/\sqrt{n_1}, \cdots, e/\sqrt{n_k})$$

The sum of squares function can be written as

$$J(W, \Pi) = trace(W^T W) - trace(X^T W^T W X)$$

so minimizing the above, means to maximize

$$\max_{X^T X = I_k} trace(X^T W^T W X)$$

Their summarization algorithm is summarized as

1. Compute k eigenvectors $V_k = [v_1, \cdots v_k]$ of $W_s(\alpha)$, corresponding to k largest eigenvalues.
2. Compute the pivoted QR decomposition of V_k^T as

   $$(V_k)^T P = QR = Q[R_{11}, R_{12}]$$

   Q is a k-by-k orthogonal matrix, R_{11} is a k-by-k upper triangular matrix, and P is a permutation matrix.
3. Compute

   $$\hat{R} = R_{11}^{-1}[R_{11}, R_{12}]P^T = [I_k, R_{11}^{-1} R_{12}]P^T$$

   The cluster assignment of each sentence will then be determined by the row index of the largest element in absolute value of the corresponding column of $\hat{R}$.

9.2 Cost-effective Outbreak

The problem of outbreak in the networks is to find the most effective way to select a set of nodes to detect a spreading process in the network. Solving this problem is receiving interest as under this setting many real-world problems can be modeled. A solution to this problem is suggested in [44], where they also apply their solution to city water pipe networks and blog networks.

In the former we have a limited budget to put some sensors at some nodes in the network so that water contaminants can be detected as quickly as possible. The latter focuses on the problem of information spread in blogspace, and the user tries to read a particular number of posts to get the most up-to-date information about a story which is propagating in the network.

More formally, in both problems, we seek to select a subset A of nodes in a graph $G = (V, E)$ which detect outbreak quickly. The outbreak starts at some node and spreads through edges. Associated with each edge there is a time that it takes for the contaminant to reach the target node. Depending on every node we select to put sensors, we achieve a certain placement score, which is a set function R, mapping every placement A to a real number $R(A)$

There is also a cost function $c(A)$ associated with each placement, and we expect the entire cost of the sensor placement does not exceed our budget, B. Sensors placed in the network can be of different types, and quality, resulting in variety of costs. Let $c(s)$ show the cost of buying a sensor s. then, the cost of the placement is $A : c(A) = \sum_{s \in A} c(s)$. The problem is then to maximize the reward subject to the fact that cost is minimized. This can be formulated as follows:

$$\max_{A \in V} R(A) \text{ subject to } c(A) \leqslant B$$

Depending on the time $t = T(i, s)$ at which we detect the outbreak in a scenario i, we incur a penalty $\pi_i(t)$. which is a function depending on the scenario. The goal is then to minimize the expected penalty over all possible scenarios:

$$\pi(A) \equiv \sum P(i)\pi_i(T(i, A))$$

where, for a placement $A \subset V$, $T(i, A)$ is minimum among all $T(i, s)$ for $s \in A$. $T(i, A)$ is the time until event i is detected by a sensors in A first. P is the probability distribution over the events and we assume it is given beforehand.

An alternative formulation for the problem is to define a penalty reduction function which is specific for every scenario, $R_i(A) = \pi(\infty) - \pi(A)$ and

$$R(A) = \sum_i P(i) R_i(A) = \pi(\emptyset) - \pi(A)$$

The penalty function has several important properties: Firstly, $R(\emptyset) = 0$. Secondly, R is nondecreasing. That means, for $A \subset B$, $R(A) \leqslant R(B)$. Thirdly, if we add a sensor to a small placement A, we improve the score at least as much as the time we add that sensor to a larger placement $B \supseteq A$. This property for a set function is called *sub-modularity*. Leskovec et, al [44] show that, for all placements $A \subseteq B \subseteq V$ and sensors $s \in V \backslash B$, the following is true,

$$R(A \cup \{s\}) - R(A) \geqslant R(B \cup \{s\}) - R(B)$$

Maximizing sub-modular functions is NP-hard in its general form. They develop the (cost-effective lazy forward selection) CELF algorithm which exploits sub-modularity to find *near-optimal* solution and works well in the case where costs are not constants, in which greedy algorithm badly fails. Their algorithm is guaranteed to achieve at least a fraction of $\frac{1}{2}(1 - 1/e)$ of the optimal solution even in the case where every node can have a different cost. As far as running time is concerned, the CELF algorithm runs 700 times as fast as the standard greedy algorithm.

9.2.1 Web Projections

Information retrieval methods usually make an assumption that documents are independent. However, an effective web search is not achievable unless the hyperlink relations between web pages are taken into consideration. *Web projection* concentrates on the creation and use of graphical properties of the web subgraph [43]. Web projection graph is a subgraph of the larger web graph projected using a seed set of documents retrieved as the result of a query. Leskovec et. al, [43] try to investigate how query search results project onto the web graph, how search result quality can be determined using properties of the projection graph, and if we can estimate the behavior of users with query reformulation given the projection graph.

They start with a query and collect the initial seed data using a search engine. Leskovec et, al [43] then project the retrieved documents on the web graph. Projecting these documents on the web graph results in an induced subgraph named query projection graph. A *Query connection graph* is then created by adding intermediate nodes to make the query projection graph connected. These connecting nodes are actually not part of the search results. Using these two graphs (i.e., query projection graph and query connection graph) they construct features to describe the topological properties of the graphs. These features are used in machine learning techniques to build predictive models.

Overall, they extract 55 features that best describe the topological properties of two graphs. These 55 features are grouped into four categories:

- Query projection graph features, which are to measure aspects of the connectivity of query projection graph.

- Query connection graph features, which are to measure aspects of the connectivity of query connection graph. These are useful in capturing relations between projection nodes and connection nodes.
- Combination features, which are composition of features from other groups.
- Query features that represent the non-graphical properties of the result set. This feature is calculated using the query text and the returned list of documents relevant to the query.

To do the projection their data contains 22 million web pages from a crawl of the web. The largest connected component is said to have 21 million nodes, while the second largest connected component contains merely several hundred nodes. The problem which is addressed in [43] is to project the results of a query on the web graph and extract attributes described above. This can then be used to learn a model that predicts a query's class. They also learn models to predict user behavior when reformulating queries.

9.3 Co-clustering

The basic idea of clustering is to extract unique content bearing words given a set of text documents. If these words are treated as features, then documents can be represented as a vector of these features. A possible representation for such e setting is a word-by-document matrix, in which a non-zero entry indicates the existence of a particular word (row) in a document (represented by column). Words can be clustered on the basis of the documents they appear in. This clustering assumes that the more two terms have co-occurrences in documents, the closer they are. This type of clustering can be useful in building statistical thesaurus and query reformulation. The problem of simultaneously clustering words and documents is discussed in [26].

The primary assumption in [26] is that word clustering has influence on document clustering, and so does document clustering on word clustering. A given word w_i belongs to the word cluster W_m if its association with the document cluster D_m is greater than its association with other document clusters. A natural measure of association of a word with a document cluster is considered to be the sum of edge weights to all documents in the cluster [26].

$$W_m = \{w_i : \sum_{j \in D_m} A_{ij} \geqslant \sum_{j \in D_l} A_{ij}\}, \forall l = 1, \cdots, k$$

This means that each word cluster is determined by the document cluster. Similarly, each document cluster is determined by a word cluster:

$$D_m = \{w_i : \sum_{j \in W_m} A_{ij} \geqslant \sum_{j \in W_l} A_{ij}\}, \forall l = 1, \cdots, k$$

According to Dhillon, the best word and document clustering would correspond to a minimum k-partitioning of the document-word bipartite graph. This clustering is performed using a spectral algorithm, and they show it works well by their experimental results on the Medline (1300 medical abstracts) and Cranfield (1400 aeronautical systems abstracts) corpora.

10 Graph Learning II

In this chapter we continue our review on some of the learning techniques that utilize graphs as their underlying framework.

10.1 Dimensionality Reduction

In the areas of artificial intelligence, and information retrieval, researchers usually encounter problems where they have to deal with high dimensional data which is naturally coming from a smaller number of dimensions. This data is actually low dimensional but is lying on a higher dimension space. The problem of dimensionality reduction focuses on representing high dimensional data into a lower dimension space. To do so Belkin and Niyogi [11] propose an algorithm that has several properties. First, their algorithm is quite simple, with few local computations. Second, the authors use the Laplace Beltrami operator in their algorithm. Third, their algorithm performs the dimensionality reduction in a geometric fashion. Forth, the use of Laplacian eigenmaps causes locality preserving which makes the algorithm insensitive against noise and outliers.

The problem of dimensionality reduction can be formulated as follows:
Given k points $x_1, \cdots, x_k$ in R^l, find a set of k points $y_1, \cdots, y_k$ in R^m such that, $m << l$, and y_i *represents* x_i. The algorithm described in [11] is summarized as follows:

1. In this step we should construct a graph by putting an edge between nodes i and j if x_i and x_j are close. The proximity can whether be decided by using the Euclidian distance of nodes, and applying a threshold, or can be set to 1 if both of the nodes are in the other's k-nearest neighbors.

2. In the second step, we try to weight the graph using a weight function. This weight function can be a heath kernel:
$$W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}}$$
or using a simpler approach. That is, set $W_{ij} = 1$ if and only if i and j are connected.

3. Compute eigenvalues and eigenvectors for the generalized eigenvector problem:
$$Lf = \lambda Df$$
Here D is diagonal weight matrix where $D_{ii} = \sum_j W_{ji}$, and $L = D - W$ is the Laplacian matrix. If $f_0, \cdots, f_{k-1}$ is the solution of the above equation, sorted by eigenvalues:
$$Lf_i = \lambda_i D f_i$$
and
$$0 = f_0 \leqslant \lambda_1 \leqslant \cdots \leqslant \lambda_{k-1}$$
Then, leave out f_0, the corresponding eigenvector for the eigenvalue 0, and use the next m eigenvectors to embed the data in a m-dimensional Euclidean space:
$$x_i \rightarrow (f_1(i), \cdots, f_m(i))$$

The authors conduct experiments on a simple synthetic example of "swiss roll" as well as an example from vision with vertical and horizontal bars in a visual field. In both cases they use the simpler weight function ($W_{ij} \in \{0, 1\}$) and show it works well for the evaluated datasets.

10.2 Semi-Supervised Learning

In this section we focus on a semi-supervised learning method using harmonic functions described in [74]. Semi-supervised learning has received great attention since labeled examples are too expensive and time consuming to create. Building a large labeled dataset requires a large effort of skilled human annotators.

The work in [74] adopts Gaussian fields over a continuous state space rather than random fields over discrete label set. The authors assume the solution is solely based on the structure of the data manifold. The framework of the solution is as follows:
Suppose $(x_1, y_1), \cdots, (x_l, y_l)$ are labeled, and $(x_{1+1}, y_{1+1}), \cdots, (x_{l+u}, y_{l+u})$ are unlabeled data points, where $l << u$, and $n = l+u$. Construct a connected graph of the data points with a weight function W:

$$w_{ij} = exp(-\sum_{d=1}^{m} \frac{(x_{id} - x_{jd})^2}{\sigma_d^2})$$

where x_{id} i the d^{Th} component of instance x_i represented as a vector $x_i \; inR^m$, and σ_i are length scale hyperparameters for each dimension. They first compute a function f on the graph and then assign labels based on that function. To assign probability distribution of functions f, they form the Gaussian field

$$p_\beta(f) = \frac{e^{-\beta E(f)}}{Z_\beta}$$

where β is an inverse temperature parameter, Z_β is the partition function, and $E(f)$ is the quadratic energy function:

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij}(f(i) - f(j))^2$$

Zhu et, al [74] show that the minimum energy function is harmonic

$$f = argmin_{f|L=f_i} E(f)$$

The harmonic threshold is used to determine the class labels using the weight function. That is, assign 1 if $f > 1/2$ and assign 0 otherwise.

Further, they describe how to use external classifiers, classifiers that are trained separately on labeled data point and are at hand. To combine the external classifier with harmonic function, for each unlabeled node i they create a dongle node with the label given by the external classifier, and assign the transition probability from i to its dongle to η, and discount all other transitions from i by $1 - \eta$. Then they perform the harmonic minimization on this augmented graph.

10.3 Diffusion Kernels

This section covers a feature extraction method described in [71]. This article uses a graph of genes, where two genes are linked whenever they catalyze two successive reactions to see if having the knowledge represented by this graph, can help improve the performance of gene function prediction. The formulation of the problem is as follows:
The discrete set X represents the set of genes, and $|X| = n$. Then the set of expression profile is a mapping $e : X \to R^p$ with p being the number of measurements. In this setting, $e(x)$ is the expression profile of gene x. The goal is to use this graph to extract features from the expression profiles.

Let F be the set of features. Each feature is simply a mapping on the set of genes to a real number, $f : X \to R$. The normalized variance of a linear feature is defined by:

$$\forall f_{e,v} \in G, V(f_{e,v}) = \frac{\sum_{x \in X} f_{e,v}(x)^2}{\|v\|^2}$$

Linear features with a large normalized variance are called *relevant*. These can be extracted using principle component analysis techniques. Additionally, if a feature varies slowly between adjacent nodes in the graph, it is called *smooth* as opposed to *rugged*. A "good" feature is both smooth and relevant.

If we can define two functions. $h_1 : F \to R^+$, and $h_2 : G \to R^+$ for smoothness and relevance respectively, then the problem is regularized as the following optimization problem:

$$\max_{(f_1, f_2) \in F_0 \times G} \frac{f_1' f_2}{\sqrt{f_1' f_1 + \delta h_1(f_1)} \sqrt{f_2' f_2 + \delta h_2(f_2)}}$$

Here δ is a parameter to control the trade-off between smoothness and relevance. The method in [71] uses the energy at high frequency by computing the Fourier transform to specify a smoothness function of a feature on a graph. Let's assume L is the Laplacian of the graph, and $0 = \lambda_1 \leqslant \cdots \leqslant \lambda_n$ are its eigenvalues, and $\{\phi_i\}$ is the orthonormal set of corresponding eigenvectors. The Fourier decomposition of any feature $f \in F$ is

$$f = \sum_{i=1}^{n} \hat{f}_i \phi_i$$

where $\hat{f}_i = \phi_i'$. The smoothness function for a feature $f \in F$ is calculated as

$$\|f\|^2_{K_\zeta} = \sum_{i=1}^{n} \frac{\hat{f}_i^2}{\zeta(\lambda_i)}$$

where ζ is a monotonic decreasing mapping, and $K_\zeta : X^2 \to R$ is defined by

$$K_\zeta(x, y) = \sum_{i=1}^{n} \zeta(\lambda_i) \phi_i(x) \phi_i(y)$$

the matrix K_ζ is positive definite as the mapping ζ only takes positive values. as i increases, λ_i increases, so $\zeta(\lambda_i)$ decreases. Subsequently, the above norm has higher a value on a feature with a lot of energy at high frequency, so can be considered as a smoothing function. The exponential function is a good example of ζ.

The relevance of a feature is also defined in [71] as

$$h_2(f_{e,v}) = \|f_{e,v}\|_H$$

where H is the reproducible kernel Hilbert space (RKHS) associated with the linear kernel $K(x, y) = e(x)'e(y)$.

10.3.1 Reformulation

If $K_1 = exp(-\tau L)$ is the diffusion kernel, and $K_2(x, y) = e(x)'e(y)$ is the linear kernel, taking $h_1(f) = \|f\|_{H_1}$ and $h_2(f) = \|f\|_{H_2}$ the problem can be expressed in its dual form as

$$\max_{(\alpha,\beta)\in F^2} \gamma(\alpha, \beta) = \frac{\alpha' K_1 K_2 \beta}{(\alpha'(K_1^2 + \delta K_1)\alpha)^{1/2}(\beta'(K_2^2 + \delta K_2)\beta)^{1/2}}$$

The above formulation is a generalization of the canonical correlation analysis known as kernel-CCA which is discussed in [7].

Results reported by the article are encouraging. The performance is shown to be above 80% for some classes, and this method seems successful on some classes which can not be learned using SVM.

11 Sentiment Analysis I

11.1 Introduction

The proceeding two sections give reviews of some papers on sentiment and opinion analysis. Researchers have been investigating the problem of automatic text categorization and sentiment analysis for the past two decades. Sentiment analysis seeks to characterize opinionated natural language text.

11.2 Unstructured Data

The emergence of Internet users, and content generating Internet applicants, have resulted in a great increase in the amount of unstructured text on the Internet. Blogs, discussion groups, forums, and similar pages are examples of such growing content. This makes the need for a sentiment analysis technique that can handle the lack of text structure.

Most of the previous work on sentiment analysis assume the pairwise independence between features used in classification. Unlike them, the work in [9] tries to propose a machine learning technique for learning predominant sentiments of on-line texts that captures dependencies among words, and to find a minimal, and sufficient set of vocabulary to do the categorization task. Xue Bai, Rema Padman and Edoardo Airoldi [9] present a two-stage Markov Blanket Classifier (MBC) that learns conditional dependencies among the words and encodes them into a Markov Blanket Directed Acyclic Graph (MB DAG) for the sentiment variable. The classifier then uses a meta-heuristic based search, named *Tabu Search (TB)*. Detecting dependencies is important, as it allows finding semantic relations between subjective words.

Before discussing the methodology, let's give some background. A *Markov Blanket (MB)* for a random variable Y is a subset Q of a set of random variables X, such that y is independent of $X\backslash Q$, and conditional on all variables in Q. Different MB DAGs that entail the same factorization for the conditional probability of Y, conditional on a set of variables X, are said to belong to the same *Markov equivalence class*. Additionally, the search used in this paper is the TS, which is a meta-heuristic strategy that helps local search heuristics explore the state space by guiding them out of local optima [35]. The basic Tabu search is simple. It starts with a solution and iteratively chooses the best move, according to a fitness function. This search method assures that solutions previously met are not revisited in the short-term.

The algorithm can be summarized as follows:
It first generates an initial MB for Y from the data. To do so, it collects variables which are within two hops of Y in terms of geographical representation: Potential parents and children plus their potential parents and children. The graph so far is undirected. To make the graph directed their method uses the rules described in [8], and then prunes the remaining undirected edges and bi-directed edges to avoid cycles, passes them to a Tabu Search, and returns the MB DAG.

They evaluate their algorithm using the data that contains approximately $29,000$ posts to the rec.arts.movies.reviews newsgroup archived at the Internet Movie Database (IMDb). To put the data into the right format, they convert the explicit ratings into one of three categories: positive, negative, or neutral. They extracted all the words that appeared in more than 8 documents, and thus were left with a total number of $7,716$ words, as input features. So each document in this experiment is represented as

$$X = [X_1, \cdots, X_{7,716}]$$

11.3 Appraisal Expressions

Appraisal expression extraction can be viewed as a fundamental task in sentiment analysis. An appraisal expression is a textual unit expressing an evaluative stance toward some target [13]. The paper that I'm going to discuss in this section tries to characterize the evaluative attributes of these textual units. The method in [13] extracts adjectival expressions. An appraisal expression, by definition, contains a source, an attitude, and a target, each represented by different attributes. As a simple example consider the following sentence:

I found the movie quite monotonous.

Here the speaker (the Source) has a *negative* opinion (quite monotonous) towards the movie (the Target). The appraisal theory is based on the following definitions:

- Attitude type: affect, appreciation, or judgment.
- Orientation: positive, or negative.
- Force: intensity of the appraisal.
- Polarity: marked, or unmarked.
- Target type: semantic type for the target.

They use a chunker to find attitude groups and targets using a pre-built lexicon, which contains head adjectives and appraisal modifiers. After that, the system links each attitude to a target. Each sentence is parsed into dependency representation, and linkages are ranked so that the paths in the dependency tree connecting words in the source to words in the target can be found. After these linkages are made, this information is used to disambiguate multiple senses that an appraisal expression may present.

Let's denote the linkage type used in a given appraisal expression by ℓ, the set of all possible linkages as L, and a specific linkage type by l. Also let's denote target type of a given appraisal expression by τ , the set of all target types by T , and a specific target type by t.

The goal is to estimate, for each appraisal expression e in the corpus, the probability of its attitude type being a, given the expressions target type t and its linkage type l.

$$P(A = a|e) = P(A = a|\tau = t, \ell = l)$$

Let the model of this probability be M,

$$P_M(A = a|\tau = t, \ell = l) = \frac{P_M(\tau = t, \ell = l|A = a)P_M(A = a)}{P_M(\tau = t, \ell = l)}$$

Assuming conditional independence yields:

$$\frac{P_M(\tau = t|A = a)P_M(\ell = l|A = a)P_M(A = a)}{P_M(\tau = t)P_M(\ell = l)}$$

Given a set of appraisal expressions E extracted by chunking and linkage detection, the goal is to find the maximum likelihood model

$$M^* = arg \max_M \prod_{e \in E} \prod_{a \in A} M(A = a|e)$$

They perform two separate evaluations on the system to evaluate the overall accuracy of the entire system, and to specifically evaluate the accuracy of the probabilistic disambiguator.

11.4 Blog Sentiment

In this section, we discuss the work in [21] that describes textual and linguistic features extracted and used in a classifier to analyze the sentiment in blog posts. The aim to see if a post is subjective, and whether it represents a good opinion or a bad one.

The training dataset used in this paper is created in two steps. First, the authors obtained the data via RSS feeds. Objective feeds are from news sites, like CNN, NPR, etc. as well as various sites about health, world, and science. Subjective feeds include content from newspaper columns, letters to an editor, reviews and political blogs. In the second step, Chesley et, al [21] manually verify each document to confirm whether it is subjective, objective, positive, or negative. The authors use three class of features to classify the sentiment. Textual features, Part-of-Speech features, and lexical semantic features. Each post is then represented as a vector of features in SVM.

Chesley et, al [21] use the online Wikipedia dictionary to determine the polarity of adjectives in the text, as adjectives in wiktionary are often defined by a list of synonyms. This is based on their assumption that wiktionary method assumes that an adjective will most likely have the same polarity as its synonym. Each blog post, in this method, is represented as a vector with count values for each feature. A binary classification is then performed for each post using a Support Vector Machine (SVM) classifier. Their hold-out experiments show that for objective and positive posts, positive adjectives acquired from Wikipedias Wiktionary play a key role in increasing overall accuracy.

11.5 Online Product Review

A large amount of web content is subjective and shows people's reviews for different products and services. The problem of analyzing reviews gives a user an aggregate view of the entire collection of opinions, and segmenting the articles into classes that can further be explored. In this section we review the method of sentiment analysis in [23], which uses a rather large dataset, and n-grams instead of unigrams. Their dataset consists of over 200k online reviews with an average of more than 800 bytes.

Three classifiers are described in [23]. The Passive Aggressive (PA) classifiers are a family of margin based online learning algorithms, and are similar to SVM. In fact, they can be viewed as online SVM. PA tries to find a hyperplane to divide the instances into two groups. The margin is proportional to an instance's distance to the hyperplane. When the classifier encounter errors, the PA algorithm utilizes the margin and changes the current classifier. Choosing PA instead of SVM to do the classification has two advantages. First, the PA algorithm follows an online learning pattern. The PA algorithm has a theoretical loss bound, which makes the performance of the classifier predictable [23].

The second classifier is the Language Modeling (LM) based classifier, which is a generative method and classifies a word, sentence, or string by calculating its probability of generation. The probability of a string $s = w_1 \cdots w_l$ is calculated as

$$P(s) = \prod_{i=1}^{l} P(w_i | w_1^{i-1})$$

where $w_i^j = w_i \cdots w_j$.
Cui et, al [23] use the Good-Turning estimation in their method, which states that if an n-gram

occurs r times, it should be treated as if it had occurred r^* times.

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

where n_r denotes the number of n-grams that occur r times in the training data.

Window classifier is the third classifier described in [23]. Window is an online learning algorithm which has been used for sentiment analysis before. This algorithm learns a linear classifier from bag-of-words of documents to predict the polarity of a review x. More formally,

$$h(x) = \sum_{w \in V} f_w c_w(x)$$

where $c_w(x) = 1$ if word w appears in review x and is 0 otherwise. Wondow uses a threshold value V to classify the reviews based on that as positive and negative.

The main contribution of this work, however, is to explore the role of n-grams as features in analyzing sentiments when $n \geqslant 3$. In their study they set $n = 6$ and extract nearly 3 million n-grams after removing those that had appeared in less than 20 reviews. They calculate ^{2} for each n-grams. Assume the following assignments:

- t: a term.
- c: a class.
- A: the number of times t and c co-occur.
- B: the number of times t occurs without c
- C: the number of times c occurs without t
- D: the number of times neither t nor c occurs
- N: the number of documents.

Then

$$X^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}$$

After calculating this score for each n-gram, The n-grams are sorted in descending order by their ^{2} scores, and the top M ranked n-grams are chosen as features for the classification procedure.

They show that high order n-grams improve the performance of the classifiers, especially on negative instances. They also claim that their observation based on large-scale dataset has never been testified before.

12 Sentiment Analysis II

In this chapter we continue our review on some novel techniques in sentiment analysis. We look at works on classification of review, and subjectivity strength.

12.1 Movie Sale Prediction

Analyzing weblogs to extract opinions is important since they contain discussions about different products, and opinionated text segments. Mishne and Glance [54] have looked at blogs to examine the correlation between sentiment and references. The first set of data that the authors use is IMDB – the Internet Movie Database. For each movie they obtained the date of its opening, as well as the gross income during the weekend. Additionally, they have a set of weblog data, from which they extract relevant posts to each movie. the relevance of a post and a movie is determined by the following criteria:

- The post should fall within the period starting a month before to a month after the date of the movie release.
- The post should contain a link to the movie's IMDB page, or the exact movie name appears in the content of the post together with one of the words, *movie, watch, see, film).*

For each post in the list of relevant ones, Mishne and Glance extracted part of text by selecting k words around the hyperlink where the movie is referred to. The value of k is various between 6 and 250. For each context, they calculate the sentiment score as described in [58]. Their analysis was carried on 49 movies, released between February, and August 2005. The polarity score of this sentiment analysis is fitted to a log-linear distribution, with the majority of scores falling within a range of 4 to 7.

To perform some experiments, they use the Pearson's correlation between some sentiment-derived metrics (number of positive contexts, number of negative contexts, total number of non-neural contexts, ratio between positive and negative contexts, the mean and variance of sentiment values), and both income per screen, and raw sales of each movie.

The experiments in [54] show that in the domain of movies, there is a high corelation between references to the movies in blog entries, and movies gross income. They also demonstrate that the number of positive reviews correlates better than raw counts in the period prior to the release of a movie.

12.2 Subjectivity Summarization

The analysis of different opinions and subjectivity has received a great amount of attention lately because of its various applications. In this section we reviewed a new approach in sentiment analysis described by Pang and Lee [59]. Two factors distinguish this work from all previous works that focused on selecting lexical features and classified sentences based on some such features. First, in this work, labels that are assigned to sentences are either subjective or objective. Second, a standard machine learning classifier is applied to make an extract.

Document polarity can be considered as a special case of text classification in which we classify sentences to sentiments rather than topic based categories. This enables us to apply machine learning techniques to solve this problem. To extract sentiments within a corpus, a system should review the sentences, and tag their *subjectivity.* Subjective sentences should further be classified into positive and negative sentences. Suppose we have n terms, $x_1, \cdots, x_n$, and we want to classify them into C_1 and C_2. This paper utilizes a mincut-based classification which uses two weight functions: $ind_j(x)$, and $assoc(x, y)$. Here, $ind_j(x)$ shows the closeness of data point x to class j and the association function shows the similarity of two data points to each other.

We would like to maximize each item's "net happiness" (i.e., its individual score or the class it is assigned to, minus its individual score for the other class). They also aim to penalize putting highly associated items into different classes. Thus, the problem reduces to an optimization one, and the goal is to minimize

$$\sum_{x \in C_1} ind_2(x) + \sum_{x \in C_2} ind_2(x) + \sum_{x_i \in C_1, x_k \in C_2} assoc(x_i, x_k)$$

To solve this minimization problem, the authors build an undirected graph G with vertices $\{v_1, \cdots, v_n, s, t\}$. Edges (s, v_i) with weights $ind_1(x_i)$ and (v_i, t) with weights $ind_2(x_i)$ are added. After that, $\binom{n}{2}$ edges (v_j, v_k) with weights $assoc(x_j, x_k)$ are added. Now, every cut in the graph G corresponds to a partition of items that have a cost equal to the partition cost and the mincut minimizes this cost. Individual scores, $ind_1(x)$ and $ind_2(x)$, are assigned with $Pr^{NB}_{sub}(x)$ and $1 - Pr^{NB}_{sub}(x)$ respectively, where $Pr^{NB}_{sub}(x)$ is Naive Bayes' estimate of the probability that the sentence x is subjective (Weights of SVM can also be used). The degree of proximity is used as the association score of the two nodes.

$$assoc(x_i, x_j) = \begin{cases} f(j-i) \cdot c & \text{if } j - i \leqslant T \\ 0 & \text{otherwise} \end{cases}$$

The function $f(d)$ specifies how the influence of proximal sentences decays with respect to the distance d. The authors used $f(d) = 1$, e^{1-d}, and $1/d^2$.

Pang and Lee [59] show that the for the Naive Bayes polarity classifier, the subjectivity extracts are shown to be more effective inputs than the original documents. The authors also conclude that employing the mincut framework can result in an efficient algorithm for sentiment analysis.

12.3 Unsupervised Classification of Reviews

This section focuses on the unsupervised approach of classifying the reviews described in [70]. The PMI-IR method is used to estimate the orientation of a phrase. This method uses the Pointwise Mutual Information (PMI) and Information Retrieval (IR) to measure the similarity of words or phrases. The sentiment extraction method in [70] is simple. First, it extracts phrases that contain adjectives and adverbs. This is possible by applying a part of speech tagger, and then extracting the phrases with specific order of tags. In the second step the semantic orientation of the extracted phrases are estimated using PMI-IR. The PMI score between two words, w_1 and w_2 is calculated as follows

$$PMI(w_1, w_2) = \log_2\Big(\frac{p(w_1 \& w_2)}{p(w_1)p(w_2)}\Big)$$

where $p(w_1, w_2)$ shows the probability that two words co-occur. this ratio is a measure of the degree of statistical dependence between the words, whose logarithm is the amount of information that we acquire about the presence of one of the words when the other is observed.

Semantic Orientation (SO) is then calculated as

$$SO(phrase) = PMI(phrase, \text{``}excellence\text{''}) - PMI(phrase, \text{``}poor\text{''})$$

With some minor algebraic manipulations:

$$SO(phrase) = \log_2 \left[\frac{hits(phrase \text{ NEAR } \text{``}excellence\text{''})hits(\text{``}poor\text{''})}{hits(phrase \text{ NEAR } \text{``}poor\text{''})hits(\text{``}excellence\text{''})} \right]$$

where $hits(query)$ is the number of hits returned for the query. Then the system classifies the review based on the average semantic orientation of the phrases.

In experiments with 410 reviews from Epinions, the algorithm reaches an average accuracy of 74%. Movie reviews seem difficult to classify, and the accuracy on movie reviews is about 66%. On the other hand, for banks and automobiles, the accuracy is 80% to 84%. Travel reviews are an intermediate case. This might suggest that in movies, the whole is not the sum of the parts, while in banks and automobiles it is.

12.4 Opinion Strength

Subjective expressions are terms or phrases that express sentiments. According to [73] the subjective strength of a word or phrase is the strength of the opinion, emotion that is expressed. Their examination of the annotated data shows that strong subjectivity is expressed in many different ways. Subjectivity clues (PREV) include words and phrases obtained from manually annotated sources. Due to the variety of clues and their sources, the set of PREV clues is not limited to a fixed word list or to words of a particular part of speech.

The syntactic clues are mostly developed by using a supervised learning procedure. The training data is based on both a human annotated (the MPQA) corpus and a large unannotated corpus in which sentences are automatically identified as subjective or objective through a bootstrapping algorithm. The learning procedure consists of three steps. First, the sentences in the training data are parsed with the Collins' parser. Second, five syntactic clues are formed for each word w, in every parse tree. The class of a word in a parse tree can be one of the following:

- $root(w, t)$: word w with POS t is the root of the parse tree.
- $leaf(w, t)$: word w with POS tag t is a leaf in the dependency parse tree.
- $node(w, t)$: word w with POS tag t.
- $bilex(w, t, r, w_c, t_c)$: word w with POS tag t is modified by word w_c with POS tag t_c, with grammatical relationship of r.
- $allkids(w, t, w_1, t_1, \cdots, r_n, w_n, t_n)$: Word w with tag t has n children, w_i with tags t_i that modify w with grammatical relationship r_i.

Some rules are set to specify the usefulness of a clue.

> A clue is considered to be potentially useful if more than x% of its occurrences in the training data are in phrases marked as subjective where x is a parameter tuned on the training data.

The authors have set $x = 70$ in their experiments. Useful clues are then put into three classes: highly reliable, not very reliable, and somewhat reliable. These reliability levels are determined by occurrence. For each clue c they calculate $P(strength(c)) = s$ as the probability of c being in an annotation of strength s. If $P(strength(c)) = s \geqslant T$ for some threshold T, c is put in the set with strength s.

Their experiments show promising results in extracting opinions in nested clauses, and classifying their strength. Syntactic clues are used as features, and the classification is done using the bootstrapping method with the 10-fold cross-validation experiments. Improvements in accuracy of the new system compared to a baseline range from 23% to 79%.

13 Spectral Methods

This last chapter covers the spectral methods in graphs. Particularly, we look at transductive learning, and spectral graph partitioning.

13.1 Spectral Partitioning

Spectral partitioning is a powerful, yet expensive technique for graph clustering [33]. This method works based on the definition of *incidence matrix*. The incidence matrix, $In(G)$, of the graph G is an $|N| \times |E|$ matrix, with one row for each node and one column for each edge. A column is zero except for two values i and j which are $+1$ and -1 respectively if there is an edge $e(i,j)$ in the graph. Additionally, the Laplacian matrix $L(G)$ of G is an $|N| \times |N|$ symmetric matrix, with one row and column for each node, where

$$(L(G))(i,j) = \begin{cases} deg(i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and there is an edge } (i,j) \\ 0 & \text{Otherwise.} \end{cases}$$

$L(G)$ has several properties:

1. Symmetry. This causes eigenvalues of $L(G)$ to be real, and its eigenvectors be real and orthogonal.
2. If $e = [1, ..., 1]^T$, then $L(G).e = 0$.
3. $In(G).(In(G))^T = L(G)$
4. If for a non-zero v, $L(G).v = lambda.v$ then,
$$\lambda = \frac{\|(In(G)^T.v)\|^2}{\|v\|^2}$$
5. $0 \leqslant \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$ where, λ_i are eigenvalues of $L(G)$.
6. The number of connected components of G is equal to the number of λ_i's that are equal to 0. This means that $\lambda_2 \neq 0$ if and only if G is connected. ($\lambda_2(L(G))$ is called the algebraic connectivity of G)

Algorithm 1 Spectral partitioning algorithm

```
1: Compute the eigenvector $v_2$ corresponding to $\lambda_2$ of $L(G)$
2: for Each node $n$ of $G$ do
3:   if $v_2(n) < 0$ then
4:     Put node $n$ in partition $N-$
5:   else
6:     Put node $n$ in partition $N+$
7:   end if
8: end for
```

It is shown in [33] that if G is connected, and $N-$ and $N+$ are defined by algorithm 1, then $N-$ is connected. It can also be shown that, if $G_1 = (N, E_1)$ is a subgraph of $G = (N, E)$, such that G_1 is "less connected" than G, then $\lambda_2(L(G_1)) \leqslant \lambda_2(L(G))$.

13.2 Community Finding Using Eigenvectors

In this section we overviewed the method of finding communities in networks using eigenvectors suggested by Newman [57]. The problem, for which a solution is suggested, tries to cluster network data. Networks have received great attention in physics and other fields as a foundation of the mathematical representation of a variety of complex systems, including biological and social systems, the Internet, the World Wide Web, and many others. There has been a lot of effort to devise algorithms to cluster graphs. However, the major problem is to develop algorithms that can be run in parallel or distributed computing systems.

If A is the adjacency matrix of the graph, in which A_{ij} is 1 if there's an edge between two vetrices i, j, then the number of edges going from one cluster to another is

$$R = \frac{1}{2} \sum_{i,j \text{ in different clusters}} A_{ij}$$

The index vector, s, defines for a vertex, whether it is in the first group or the second. It assigns a vertex a value of $s_i = 1$ in the former case, and $s_i = -1$ in the latter. The following is the immediate result of the index vector's definition.

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1 & \text{if } i, j \text{ are in different groups} \\ 0 & \text{if } i, j \text{ are in the same groups} \end{cases}$$

Then

$$R = \frac{1}{4} \sum_{ij} (1 - s_i s_j) A_{ij}$$

Also the degree of node i, k_i, is calculated as

$$k_i = \sum_j A_{ij}$$

The above equations can be reformulated in the matrix form as

$$R = \frac{1}{4} s^T L s$$

where $L_{ij} = k_i \delta_{ij} - A_{ij}$ and δ_{ij} is 1 if $i = j$ and 0 otherwise. L is called the *Laplacian matrix* of the graph. Assume λ_i is the eigenvalue of L corresponding to the eigenvector v_i, and that $\lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_n$. Furthermore, if n_1, n_2 are the required sizes of groups, then

$$a_1^2 = (v_1^T s)^2 = \frac{(n_1 - n_2)^2}{n}$$

where $a_i = v_i^T s$.

Newman [57] defines the notion of modularity of a graph Q as $Q =$ (number of edges within communities) $-$ (expected number of such edges). The goal of network clustering is then reduced to the problem of optimizing the modularity of the network. More formally, Q can be defined as

$$Q = \frac{1}{2m} \sum_{ij} [A_{ij} - P_{ij}] \delta(g_i, g_j)$$

where P_{ij} is the expected number of the edges falling between a particular pair of vertices, i and j. In this setting, A_{ij} is the actual number of such edges, and $\delta(r, s) = 1$ if $r = s$ and 0 otherwise. Note that $\delta(g_i, g_j) = \frac{1}{2}(s_i s_j + 1)$ thus,

$$Q = \frac{1}{4} \sum_{ij} [A_{ij} - P_{ij}](s_i s_j + 1)$$

and the matrix form

$$Q = \frac{1}{4m} s^T B s$$

where $B_{ij} = A_{ij} - P_{ij}$. B is called the *modularity matrix.* He solves this equation as

$$Q = \frac{1}{4m} \sum_i \alpha_i^2 \beta_i$$

where β_i is the eigenvalue of B corresponding to eigenvector u_i. Thus, Newman shows that the modularity of a network can be represented using eigenvalues and eigenvectors of a matrix and is called the modularity matrix. Using this expression Newman [57] derives algorithms for identifying communities, and detecting bipartite or k-partite structure in networks, and a new community centrality measure.

13.3 Spectral Learning

Spectral learning techniques are algorithms that use information contained in the eigenvectors of a data affinity matrix to detect structure. The method described in [40] is summarized as follows:

1. Given data B, the affinity matrix is $A \in R^{n \times n} = f(B)$.
2. Form D, a diagonal matrix, with $D_{ii} = \sum_j A_{ij}$.
3. Normalize: $N = (A + d_{max} I - D)/d_{max}$.
4. Let $x_1, \cdots, x_k$ be the k largest eigenvectors of N and form a normalized matrix using $X = [x_1, \cdots, x_k] \in R^{n \times k-1}$
5. Assign a data point x_i to j if and only if the row X_i is assigned to j.

Kamvar et, al [40] specify a data-to-data Markov transition matrix. If A is the similarity matrix of the documents, then the equality $A = B^T B$ holds for a term-document matrix B. To map document similarities to transition probabilities, let's define $N = D^{-1} A$, where D is a diagonal matrix with $D_{ii} = \sum_j D_{ij}$. Here, N corresponds to a transitioning with probability proportional to relative similarity values.

Four different datasets are used in [40] to compare the spectral learning algorithm to K-means: 20 newsgroups, 3 newsgroups, LYMPHOMA, SOYBEAN.

They further suggest an algorithm for classification.

1. Define A as previously mentioned.
2. For each pair of points (i, j), if they are in the same class, assign $A_{ij} = A_{ji} = 1$.

3. For each pair (i, j) if they're in different classes, set $A_{ij} = A_{ji} = 0$.
4. Normalize $N = \frac{1}{d_{max}}(A + d_{max}I - D)$

If natural classes occur in the data, the Markov chain described above should have cliques. The key differences between the spectral classifier and the clustering algorithm are that A incorporates labeling information, and a classifier is used in the spectral space rather than a clustering method. This algorithm is able to classify documents by the similarity of their transition probabilities to known subsets of B, and this makes this method novel.

1. Given the data B, the affinity matrix is $A \in R^{n \times n} = f(B)$.
2. Form D, a diagonal matrix, with $D_{ii} = \sum_j A_{ij}$.
3. Normalize: $N = (A + d_{max}I - D)/d_{max}$.
4. Let $x_1, \cdots, x_k$ be the k largest eigenvectors of N and form a normalized matrix using $X = [x_1, \cdots, x_k] \in R^{n \times k-1}$,
5. Represent each data i by the row X_i of X.
6. Classify the rows as points using a classifier.
7. Assign a data point i to the class that X_i is assigned.

13.4 Transductive learning

Transductive learning is referred to some applications, in which the examples are already known when a classifier is trained. Relevance feedback is an example of such tasks, in which users give positive and negative labels to examples that are in the training set.

The learning task is defined on a fixed array X of n points $(\vec{x_1}, \vec{x_2}, \cdots, \vec{x_n})$. The classification label for each data point is denoted by y_i and can either be $+1$ or -1. A transductive learner can analyze the location of all points and so can structure its hypothesis space based on the input. The method of spectral graph transducer (SGT) is described in [39]. This algorithm takes as input the training labels Y_l, and a weighted undirected graph on X with adjacency matrix A. The similarity-weighted k-nearest neighbor graph can be made as

$$A'_{ij} = \begin{cases} \frac{sim(\vec{x_i}, \vec{x_j})}{\sum_{\vec{x_k} \in knn(\vec{x_k})} sim(\vec{x_i}, \vec{x_k})} & \text{if } x_j \in knn(\vec{x_i}) \\ 0 & \text{Otherwise} \end{cases}$$

First, the diagonal degree matrix B should be computed, in which $B_{ii} = \sum_j A_{ij}$. Then, the Laplacian matrix can be computed, $L = B - A$. The normalized Laplacian is computed as

$$L = B^{-1}(B - A)$$

The smallest $d + 1$ eigenvalues (excluding the first) and their corresponding eigenvectors of L are chosen and assigned to D and V respectively. For each new training set:

- $\hat{\gamma}+ = \sqrt{\frac{l_-}{l_+}}$ and $\hat{\gamma}- = \sqrt{\frac{l_+}{l_-}}$

- Set $C_{ii} = \frac{l}{2l_+}$ for positive examples and $C_{ii} = \frac{1}{2l_-}$ for negative examples, where C is the cost of training samples. This will guarantee equal costs of positive and negative samples.
- Compute $G = (D + cV^TCV)$ and $\vec{b} = cV^TC\vec{\gamma}$. Find λ^*, the smallest eigenvalue of

$$\begin{bmatrix} G & -I \\ -\frac{1}{n}\vec{b}\vec{b}^T & g \end{bmatrix}$$

- predictions can be computed as

$$\vec{z}^* = V(G - \lambda^* I)^{-1}\vec{b}$$

Finally, to do the hard class assignment we can threshold $\vec{z}^*$ wrt. $\frac{1}{2}(\hat{\gamma}_+ + \hat{\gamma}_-)$, and set $y_i = sign(z_i - \frac{1}{2}(\hat{\gamma}_+ + \hat{\gamma}_-))$

References

[1] L. Abderrafih. Multilingual alert agent with automatic text summarization.

[2] Lada Adamic and Natalie Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proceedings of the WWW2005 Conference's 2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis, and Dynamics*, 2005.

[3] Eytan Adar, Li Zhang, Lada A. Adamic, and Rajan M. Lukose. Implicit structure and the dynamics of Blogspace. 2004.

[4] S. Afantenos, V. Karkaletsis, and P. Stamatopoulos. Summarization from medical documents: a survey. *Artificial Intelligence In Medicine*, 33(2):157–177, 2005.

[5] David J. Aldous and James A. Fill. *Reversible Markov Chains and Random Walks on Graphs.* Book in preparation, `http://www.stat.berkeley.edu/~aldous/book.html`, 200X.

[6] L. Antiqueira, MGV Nunes, ON Oliveira Jr, and L.F. Costa. Complex networks in the assessment of text quality. *Arxiv preprint physics/0504033*, 2005.

[7] Francis R. Bach and Michael I. Jordan. Kernel independent component analysis. *J. Mach. Learn. Res.*, 3:1–48, 2003.

[8] Glymour C Padman R. Spirtis P. Bai, X. and J. Ramsey. Mb fan search classifier for large data sets with few cases. In *Technical Report CMU-CALD-04-102. School of Computer Science, Carnegie Mellon University*, 2004.

[9] Xue Bai, Rema Padman, and Edoardo Airoldi. Sentiment extraction from unstructured text using tabu search-enhanced markov blanket. In *Workshop on Mining the Semantic Web, 10th ACM SIGKDD Conference*, 2004.

[10] R. Barzilay and M. Elhadad. Using Lexical Chains for Text Summarization. *Advances in Automatic Text Summarization*, 1999.

[11] Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[12] R. Blood. How blogging software reshapes the online community. *Communications of the ACM.*, 47:53–55, 2004.

[13] Kenneth Bloom, Navendu Garg, and Shlomo Argamon. Extracting appraisal expressions. In *HLT-NAACL 2007*, pages 308–315, 2007.

[14] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 19th International Conference on Machine Learning (ICML-2001)*, 2001.

[15] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. pages 19–26, 2001.

[16] Avrim Blum, John D. Lafferty, Mugizi Robert Rwebangira, and Rajashekar Reddy. Semi-supervised learning using randomized mincuts. 2004.

[17] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. In *Proceedings of the International Conference on Computer Vision (ICCV 1)*, pages 377–384, 1999.

[18] Jaime G. Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. pages 335–336, 1998.

[19] Jean Carletta. Assessing agreement on classification tasks: the kappa statistic. *Comput. Linguist.*, 22(2):249–254, 1996.

[20] H.H. Chen and C.J. Lin. A multilingual news summarizer. *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 159–165, 2000.

[21] Paula Chesley, Bruce Vincent, Li Xu, and Rohini Srihari. Using verbs and adjectives to automatically classify blog sentiment. In *Proceedings of AAAI-CAAW-06, the Spring Symposia on Computational Approaches to Analyzing Weblogs*, 2006.

[22] N. Chomsky. *Syntactic Structures.* Walter de Gruyter, 2002.

[23] Hang Cui, Vibhu O. Mittal, and Mayur Datar. Comparative experiments on sentiment classification for online product reviews. In *AAAI*, 2006.

[24] H. Dalianis, M. Hassel, K. de Smedt, A. Liseth, TC Lech, and J. Wedekind. Porting and evaluation of automatic summarization. *Nordisk Sprogteknologi*, 1988.

[25] G. DeJong. An Overview of the FRUMP Sy ste m. *Strategies for Natural Language Processing*, 1982.

[26] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, New York, NY, USA, 2001. ACM.

[27] Sergey N. Dorogovtsev and José Fernando F. Mendes. Language as an evolving word Web. *Proceedings of the Royal Society of London B*, 268(1485):2603–2606, December 22, 2001.

[28] P. Doyle and J. Snell. *Random walks and electric networks.* Math. Assoc. America., Washington, 1984.

[29] HP Edmundso. New Methods in Automatic Extracting. *Advances in Automatic Text Summarization*, 1999.

[30] Güneş Erkan and Dragomir R. Radev. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*, 2004.

[31] Ramon Ferrer i Cancho and Ricard V. Solé. The small-world of human language. *Proceedings of the Royal Society of London B*, 268(1482):2261–2265, November 7 2001.

[32] Ramon Ferrer i Cancho, Ricard V. Solé, and Reinhard Köhler. Patterns in syntactic dependency networks. 69(5), May 26, 2004.

[33] Notes for Lecture 23 April 9 Berkeley. Graph partitioning, 1999. http://www.cs.berkeley.edu/ demmel/cs267/lecture20/lecture20.html.

[34] ML Glasser and IJ Zucker. Extended Watson Integrals for the Cubic Lattices. *Proceedings of the National Academy of Sciences of the United States of America*, 74(5):1800–1801, 1977.

[35] Fred Glover and Fred Laguna. *Tabu Search.* Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[36] Daniel Gruhl, R. Guha, David Liben-Nowell, and Andrew Tomkins. Information diffusion through blogspace. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 491–501, New York, NY, USA, 2004. ACM.

[37] David Hull. Using statistical testing in the evaluation of retrieval experiments. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 329–338, New York, NY, USA, 1993. ACM.

[38] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[39] Thorsten Joachims. Transductive learning via spectral graph partitioning. pages 290–297, 2003.

[40] Sepandar D. Kamvar, Dan Klein, and Christopher D. Manning. Spectral learning. pages 561–566, 2003.

[41] Klaus Krippendorff. *Content Analysis: An Introduction to Its Methodology.* Sage Publications, London, 1980.

[42] Ravi Kumar, Jasmine Novak, Prabhakar Raghavan, and Andrew Tomkins. On the bursty evolution of blogspace. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 568–576, New York, NY, USA, 2003. ACM.

[43] Jure Leskovec, Susan Dumais, and Eric Horvitz. Web projections: learning from contextual subgraphs of the web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 471–480, New York, NY, USA, 2007. ACM.

[44] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, New York, NY, USA, 2007. ACM.

[45] Y. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. Tseng. Discovery of blog communities based on mutual awareness. In *Proceedings of the WWW06 Workshop on Web Intelligence*, 2006.

[46] HP Luhn. The Automatic Creation of Literature Abstracts. *Advances in Automatic Text Summarization*, 1999.

[47] M Maier M Hein. Manifol denoising. *Advances in Neural Information Processing Systems (NIPS)*, 2006],.

[48] I. Mani and E. Bloedorn. Summarizing Similarities and Differences Among Related Documents. *Advances in Automatic Text Summarization*, 1999.

[49] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[50] D. Marcu. The rhetorical parsing of natural language texts. *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 96–103, 1997.

[51] D. Marcu. *The Theory and Practice of Discourse Parsing and Summarization.* MIT Press, 2000.

[52] A. Mehler. Compositionality in quantitative semantics. a theoretical perspective on text mining. *Aspects of Automatic Text Analysis, Studies in Fuzziness and Soft Computing, Berlin. Springer*, 2006.

[53] I.A. Mel'čuk. *Dependency Syntax: Theory and Practice.* State University of New York Press, 1988.

[54] Gilad Mishne and Natalie Glance. Predicting movie sales from blogger sentiment. In *AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW 2006)*, 2006.

[55] T. Mitchell. The role of unlabeled data in supervised learning. In *Proc. of 6th International Symposium on Cognitive Science (invited paper)*, San Sebastian, Spain, 1999.

[56] M. Módolo. *SuPor: um Ambiente para a Exploração de Métodos Extrativos para a Sumarização Automática de Textos em Português.* PhD thesis, Dissertação de Mestrado. Departamento de Computação, UFSCar. São Carlos-SP, 2003.

[57] Mark J. Newman. Finding community structure in networks using the eigenvectors of matrices, 2006. http://arxiv.org/abs/physics/0605087.

[58] Kamal Nigam and Matthew Hurst. Towards a robust metric of opinion. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, 2004.

[59] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL2004*, pages 271–278. Association for Computational Linguistics, 2004.

[60] T.A.S. Pardo and L.H.M. Rino. Descrição do GEI-Gerador de Extratos Ideais para o Português do Brasil. *Série de Relatórios do NILC NILC-TR-04-07, Núcleo Interinstitucional de Lingüística Computacional (NILC), São Carlos-SP*, 8.

[61] T.A.S. Pardo and L.H.M. Rino. GistSumm: A Summarization Tool Based on a New Extractive Method. *Computational Processing of the Portuguese Language: Proceedings*, 2003.

[62] Thiago Alexandre Salgueiro Pardo, Lucas Antiqueira, Maria das Graças Volpe Nunes, Osvaldo N. Oliveira Jr., and Luciano da Fontoura Costa. Modeling and evaluating summaries using complex networks. In *Proceedings of Computational Processing of the Portuguese Language, the Seventh International Workshop (PROPOR '06)*, pages 1–10. Springer, 2006.

[63] G. Pólya. Über eine Aufgabe der Wahrscheinlichkeitsrechnung betreffend der Irrfahrt im Straßennetz. *Math. Annalen*, 84:149–160, 1921.

[64] D.R. Radev and K. McKeown. Generating Natural Language Summaries from Multiple On-Line Sources. *Computational Linguistics*, 24(3):469–500, 1998.

[65] H. Saggion and G. Lapalme. Generating indicative-informative summaries with sumUM. *Computational Linguistics*, 28(4):497–526, 2002.

[66] G. Salton, A. Singhal, M. Mitra, and C. Buckley. AUTOMATIC TEXT STRUCTURING AND SUMMARIZATION. *Advances in Automatic Text Summarization*, 1999.

[67] Ricard V. Solé, Bernat Corominas Murtra, Sergi Valverde, and Luc Steels. Language networks: Their structure, function and evolution. Technical Report 05-12-042, Santa Fe Institute Working Paper, 2005.

[68] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2001. MIT Press.

[69] E. M. Trevino. Blogger motivations: Power, pull, and positive feedback. In *Internet Research 6.0*, 2005.

[70] Peter Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *acl2002*, pages 417–424, 2002.

[71] Jean-Philippe Vert and Minoru Kanehisa. Graph-driven features extraction from microarray data using diffusion kernels and kernel CCA. 2002.

[72] Duncan J. Watts and Steven Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, June 1998.

[73] Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. Just how mad are you? finding strong and weak opinion clauses. In *aaai2004*, pages 761–766, 2004.

[74] X. Zhu and Z. Ghahramani and J. Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of International Conference on Machine Learning*, 2003.

[75] Hongyuan Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 113–120, New York, NY, USA, 2002. ACM.

[76] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. Technical Report MPI no. 112, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, June 2003.