UIUC/MUSC at TREC 2005 Genomics Track

ChengXiang Zhai[1], Xinghua Lu[2], Xu Ling[1], Xin He[1], Atulya Velivelli[3],
Xuanhui Wang[1], Hui Fang[1], Azadeh Shakery[1]

[1]Department of Computer Science
University of Illinois at Urbana-Champaign

[2]Dept of Biostatistics, Bioinformatics and Epidemiology
Medical University of South Carolina

[3]Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

Abstract

We report experiment results from the collaborative participation of UIUC and MUSC in the TREC 2005 Genomics Track. We participated in both the adhoc task and the categorization task, and studied the use of some mixture language models in these tasks. Experiment results show that a structured theme-based language modeling approach is effective in improving retrieval effectiveness for the ad hoc taks and the Latent Dirichlet Allocation method is effective in dimension reduction for the categorization task.

1 Introduction

The University of Illinois at Urbana-Champaign (UIUC) and Medical University of South Carolina (MUSC) collaborated on both tasks of TREC 2005 Genomics Track with UIUC being more focused on the first ad hoc task while MUSC more on the second categorization task. Our general goal was to explore the effectiveness of several language modeling approaches in these tasks. Specifically, for the ad hoc task, our goal was to study a structured theme-based language model and use it to perform both pseudo feedback and relevance feedback. For the categorization task, our goal was to study the effectiveness of using the Latent Dirichlet Allocation [1] to perform dimension reduction to alleviate the problem of data sparseness.

Experiment results show that these approaches are quite promising. The structured theme-based language modeling approach provides a natural way of performing feedback for the structured queries and is shown to improve retrieval accuracy for the ad hoc task. For the categorization task, the Latent Dirichlet Allocation method is shown to be effective in dimension reduction. Compared with other participating groups, our ad hoc runs are mostly above the median and are the best for a few topics, while our categorization runs are mostly below the median due to a deficiency of our basic categorization method (SVM), which prevented our system from achieving a full-spectrum of precision-recall tradeoff.

2 Ad Hoc Retrieval Task

One interesting characteristic of the ad hoc retrieval task of the TREC 2005 Genomics Track is that the queries are structured; each query is an instantiation of a template with entities such as diseases and genes. It is thus very interesting to study whether such additional information about query structures can be exploited to improve retrieval. In particular, we are interested in studying how to extend the language models developed for unstructured text queries to handle such structured queries. We propose a structured theme-based language model to combine the information from multiple fields of a query and study how we can estimate such a language model with feedback documents. Experiment results show that such a new language model is effective for both pseudo feedback and relevance feedback.

2.1 KL-divergence retrieval model

Our basic retrieval method is the Kullback-Leibler (KL) divergence retrieval model [6, 8]. According to this method, given a query Q and a document D, we estimate a query language model θ_Q and a document language model θ_D, and then simply score the document using the KL-divergence of θ_Q and θ_D, defined as

$$s(Q,D) = D(\theta_Q||\theta_D) = \sum_{w \in V} p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_D)}$$

where V is the vocabulary set. In practice, for the sake of efficiency, we truncate our query model θ_Q and only use a certain number (100 in our experiments) of highest probability words for scoring.

Clearly, using such a method, our main tasks are to estimate θ_Q and θ_D. We estimate θ_D using Dirichlet prior smoothing, i.e.,

$$p(w|\theta_D) = \frac{c(w,D) + \mu p(w|C)}{|D| + \mu}$$

where $p(w|C)$ is a collection background language model and μ is a smoothing parameter, which we empirically set to 100 based on some tuning with the 10 training topics available to us.

The main research question we address is how to estimate θ_Q for a structured query. When the queries are unstructured, we can use the relative frequency counts of terms in the query to estimate θ_Q or use feedback documents to update this model by interpolating it with another topic language model estimated based on the feedback documents [8]. We propose a theme-based language model, which would allow us to estimate a query language model for structured queries based on either the original structured query or feedback documents.

2.2 Theme-based query language models

In order to use the KL-divergence method to score a document w.r.t. a structured query, our general idea is to model each query field with a unigram language model and define the overall query language model as a mixture model with each field model as a component. Formally, let $Q = (Q_1, ..., Q_k)$ be a structured query with k fields. Q_i is the i-th query field. For example, query 116 "Provide information about the role of the gene Insulin receptor gene in the disease Cancer" is represented as 3 fields based on the markup in the topic description: Q_1="Provide information about the role of the gene"; Q_2="insulin receptor gene"; Q_3="cancer" [1]

Intuitively, each query field characterizes one aspect of the user's information need, thus can be modeled using a theme unigram language model. Thus, we have k unigram language models for query Q, θ_1, ..., θ_k, with θ_i modeling field Q_i. The overall information need of the user can then be defined as a combination of these field language models:

$$p(w|\theta_Q) = \sum_{i=1}^{k} \pi_i p(w|\theta_i)$$

where π_i is the weight on field model θ_i. The remaining questions are how to estimate each theme model θ_i and the mixing weights π_i's.

2.3 Estimating a theme-based query model

The simplest method for estimating a theme-based query model is: (1) Treat each query field as a sample of words drawn from a multinomial distribution (i.e., a theme model) and estimate the underlying theme model using the maximum likelihood estimator, giving us $p(w|\theta_i) = p(w|Q_i) = \frac{c(w,Q_i)}{|Q_i|}$. (2) Set π_i's to be uniform, i.e., $\pi_i = 1/k$. We call this method **ThemeOrig**, which can be regarded as the baseline theme language model approach.

Since some fields in the query are biological entities (e.g., genes), one natural way to improve our query model is to automatically perform query expansion using resources such as LocusLink and MeSH. In particular, we used LocusLink to add additional gene synonyms for any given gene name in the query. To avoid introducing too much noise, we only include gene symbols if their official full names match the gene in the query well. This is a conservative, but relatively reliable way of expansion. For MeSH, we look up each query term in the MeSH term list. For each candidate MeSH term matched, we check if the term is entirely contained in our query, and if so, we add its other equivalent terms in the MeSH database to the query.

With such field expansion, we obtain an expanded structured query $Q_E = (Q_1, ..., Q_k, E_1, ..., E_k)$, where E_i is the expanded text for field Q_i. For example, if Q_i is a gene, E_i would be a list of gene symbols found from LocusLink. Naturally, some E_i's are empty. We can now obtain a presumably improved estimate of θ_i by combining E_i with Q_i:

$$p(w|\theta_i) = p(Q_i|\theta_i)p(w|Q_i) + p(E_i|\theta_i)p(w|E_i)$$

[1] In our experiments, we excluded the background field, as some preliminary experiments showed that excluding it improves performance.

$$= \quad p(Q_i|\theta_i)\frac{c(w,Q_i)}{|Q_i|} + p(E_i|\theta_i)\frac{c(w,E_i)}{|E_i|}$$

where $c(w,Q_i)$ and $c(w,E_i)$ are the counts of word w in Q_i and E_i, respectively and $|Q_i|$ and $|E_i|$ are the lengths of Q_i and E_i, respectively. We call such a method **ThemeExp**.

The probabilities $p(Q_i|\theta_i)$ and $p(E_i|\theta_i)$ are the weights on the original query field and the expanded field, and clearly, $p(Q_i|\theta_i) + p(E_i|\theta_i) = 1$. In our experiments, we empirically set both $p(Q_i|\theta_i)$ and $p(E_i|\theta_i)$ to 0.5, because our focus was on experimenting with improving the estimation through feedback documents. Unfortunately, later we found that the 0.5 setting is far from optimal and has hurt one of our official runs significantly. Indeed, when $|E_i| << |Q_i|$, such a setting would give an expanded term a substantially higher weight than terms in the original query field. How to optimize these weights is an interesting question that we should explore in the future.

A major research question we address in our TREC experiments is how we can further improve such a structured language model through both pseudo feedback and relevance feedback. We now propose a mixture model approach to use a set of feedback documents to improve our estimation of both the theme language models θ_i's and the mixing weights π_i's. Specifically, suppose we have a set of theme models θ_i's which are estimated based on the original query text and/or expanded query text. We assume that our feedback documents $F = \{D_1, ..., D_m\}$ are sampled from a mixture model with k component multinomial distributions (corresponding to the k fields of the query) and mixing distribution π_i's. We use θ_i to define a Dirichlet prior for the i-th component multinomial distribution, so that each component model will be biased to model the corresponding original query field. We also introduce a special background language model θ_0 to model the general English words and any non-relevant information in F. Our parameters for such a model can thus be represented as $\Lambda = \{\phi_i, \pi_i\}_{i=0}^{k}$, where ϕ_0 is the background model; ϕ_1, ..., ϕ_k are the k query theme language models; and π_i's are the mixing weights. The parameters can be estimated using the Maximum A Posterior (MAP) estimator:

$$\Lambda* \quad = \quad argmax_\Lambda p(F|\Lambda)p(\Lambda)$$

where $p(\Lambda) \propto \prod_{j=0}^{k}\prod_{w\in V} p(w|\phi_j)^{\sigma_j p(w|\theta_j)}$ is a conjugate prior on all the component multinomial distributions and $p(F|\Lambda)$ is the likelihood of the feedback documents given by

$$p(F|\Lambda) = \prod_{i=1}^{m}\prod_{w\in V}[\sum_{j=0}^{k}\pi_{D_i,j}p(w|\phi_j)]^{c(w,D_i)}$$

The MAP estimate can be found using the EM algorithm [3]. The updating formulas are as follows:

$$\begin{aligned}
p^{(n+1)}(z_{w,i}=j) &= \frac{\pi^{(n)}_{D_i,j}p^{(n)}(w|\phi_j)}{\sum_{j'=1}^{k}\pi^{(n)}_{D_i,j'}p^{(n)}(w|\phi_{j'})} \\
\pi^{(n+1)}_{D_i,j} &= \frac{\sum_{w\in V}c(w,D_i)p^{(n+1)}(z_{w,i}=j)}{|D_i|} \\
p^{(n+1)}(w|\phi_j) &= \frac{\sum_{i=1}^{m}c(w,D_i)p^{(n+1)}(z_{w,i}=j)+\sigma_j p(w|\theta_j)}{\sum_{w'\in V}\sum_{i=1}^{m}c(w',D_i)p^{(n+1)}(z_{w',i}=j)+\sigma_j}
\end{aligned}$$

The parameters σ_j's specify our confidence in the prior, which, in effect, control the amount of expansion. A larger σ_j would cause a smaller amount of expansion. In our experiments, we empirically set $\sigma_j = 10m$ for $j = 1, ..., k$ (for the component models) and $\sigma_0 = 100m$ (for the background model), where m is the total number of feedback documents. Parameterizing σ_j with m allows us to interpret σ_j as an "equivalent sample of text" comparable with each document. For example, when $\sigma_j = 10m$, the strength of our prior is roughly equivalent to 10 words in a feedback document.

Once we estimate ϕ_i's and $\pi_{D_i,j}$'s, we can compute the query model as

$$\begin{aligned}
p(w|\theta'_Q) &= \sum_{j=1}^{k}p(w|\phi_j)p(\phi_j|F) \\
&\propto \frac{1}{k}\sum_{j=1}^{k}p(w|\phi_j)\sum_{i=1}^{m}\pi_{D_i,j}
\end{aligned}$$

We call such a method **ThemeFB**

2.4 Experiment Results

We used the Lemur toolkit (http://www.lemurproject.org/) to do all our experiments. Preprocessing of documents is minimum and mainly involves normalization of potential gene names. We noticed the variations in gene name spelling caused by various ways of separating name constituents using white spaces, hyphens, slashes and brackets. To deal with these variations, we use a tokenizer to convert the input text into a sequence of tokens, where each token is either a sequence of lowercase letters or a sequence of numbers. White spaces and all other symbols are treated as token delimiters. For instance, the different synonyms for gene *cAMP dependent protein kinase 2*, "PKA C2", "Pka C2", and "Pka-C2", would all be normalized to the same token sequence "pka c 2" to allow them to match each other.

We submitted two official runs – UIUCgAuto and UIUCgInt. For both runs, we first use the baseline KL-divergence method **Baseline** (i.e., ignoring the structures in the query and treating it as an unstructured text query) to obtain initial retrieval results. For UIUCgAuto, we use the top 5 documents from **Baseline** to fit the mixture theme language model with a prior defined using a theme language model computed based on the original query fields (i.e., **ThemeOrig**). It thus represents a completely automatic run without using any biological resources or having any human interaction. For UIUCgInt, we asked two people with some biology background to manually judge the relevance of the top 20 documents from **Baseline** and then fit the mixture theme language model to only the judged relevant documents with a prior defined using a theme language model computed based on both the original query text and the expanded text (i.e., **ThemeExp**). This run thus attempts to benefit from both biological resources and human relevance judgments [2].

We were expecting UIUCgInt to be much better than UIUCgAuto especially since the documents we used for feedback in the case of UIUCgInt are manually judged. But surprisingly, our manual interactive/manual run UIUCgInt is worse than our automatic run! UIUCgInt has a MAP of 0.2487 while UIUCgAuto is 0.2577. A per-topic comparison shows that UIUCgAuto is better than UIUCgInt for 26 topics and worse for 23 topics out of the 49 topics that have relevant documents in the collection.

UIUCgInt differs from UIUCgAuto in two aspects: (1) theme language model prior (**ThemeOrig** for UIUCgAuto while **ThemeExp** for UIUCgInt); and (2) feedback documents (top 5 documents for UIUCgAuto while judged relevant documents for UIUCgInt). Since normally, relevance feedback is expected to help significantly, our results suggest that the LocusLink and MeSH expansion is ineffective, i.e., **ThemeExp** is worse than **ThemeOrig**, which is confirmed in our post-TREC experiments. It turns out that the problem has to do with the non-optimal weighting of the original text and the expanded text; assigning equal weights (0.5) can be significantly biased toward favoring expanded terms in some cases. This is shown in Table 1, in which we compare **ThemeExp**, **ThemeOrig**, and **Baseline**. We see that **ThemeExp** is clearly worse than **ThemeOrig** by all measures, indicating that the expanded text using LocusLink and MeSH is either mostly noise, or more likely, not weighted appropriately. Additional experiments need to conducted to further clarify this. We also see that **ThemeOrig** has higher "front-end" precision (i.e., precision at low recall levels) than **Baseline**, suggesting that constructing the query model by giving equal weights to all the fields can help increase precision, though it also hurts recall and thus the average precision.

Table 1: Comparison of Baseline, ThemeOrig, and ThemeExp

	MAP	Pr@0.1	pr@10	RelRet
Baseline	0.2415	0.447	0.382	3340
ThemeOrig	0.2366	0.459	0.398	3156
ThemeExp	0.2221	0.431	0.351	3101

We now examine the effectiveness of **ThemeFB** by comparing the feedback runs with the corresponding baseline runs in Table 2. In each run of **ThemeFB**, the prior model used is given within the parentheses. A **ThemeFB** run can thus be compared with both the **Baseline** run and the prior run. From Table 2, we see that all the feedback runs (both pseudo feedback and relevance feedback) perform better than the corresponding prior runs as well as the baseline run. Moreover, comparing relevance feedback with pseudo feedback, we see that relevance feedback performs better by all measures, as expected. These results show that the proposed theme-based language models are effective in exploiting feedback documents to improve the estimation of the query language model.

Table 2: Effectiveness of mixture theme language models

Method	MAP	Pr@0.1	pr@10	RelRet
Baseline	0.2415	0.447	0.382	3340
ThemeOrig	0.2366	0.459	0.398	3156
Pseudo ThemeFB (+ThemeOrig) = UIUCgAuto	0.2577	0.482	0.412	3476
ThemeExp	0.2221	0.431	0.351	3101
Rel ThemeFB (+ThemeExp) = UIUCgInt	0.2487	0.485	0.422	3403
Rel ThemeFB (+ThemeOrig)	0.2704	0.513	0.431	3500

Compared with other groups' submissions, both of our runs are above the median of the same type of runs [3] for a majority of topics (shown in Table 3). Moreover, UIUCgAuto is the best for 2 topics and UIUCgInt is the best for 7 topics.

[2] 5 topics have no relevant documents in the top 20 documents being judged; for these topics, we simply used the baseline results when submitting UIUCgInt.

[3] UIUCgAuto is compared with all automatic runs, while UIUCgInt all manual/interactive runs.

Table 3: Comparison with group median.

	Number of topics	
	UIUCgAuto	UIUCgInt
$>$ median	32	39
= median	0	3
$<$ median	17	7
=best	2	7

3 Categorization Task

For the text categorization task, our goal was to study the effectiveness of using the Latent-Dirichlet Allocation (LDA) for dimension reduction. Our basic categorization method is Support Vector Machine (SVM), and we mainly explored two techniques for improving its performance: (1) the use of LDA to generate semantically enriched feature representation of a document; (2) the use of a semi-supervised learning method to augment the training data. Our experiment results show that the semantic feature representation generated using LDA significantly enhances the performance of SVM and semi-supervised learning further improves the recall on the categories with very few training cases. When applied to the test data, our system had good balanced performance on all categories in terms of F values. However, the recall of the SVM was somehow capped, resulting in relatively unsatisfactory utility scores.

3.1 Support Vector Machine

Support vector machine is a well studied kernel-based classification algorithm, which searches for a linear decision surface that has the largest margin between positive and negative training data. We used the publicly available implementation, SVMlight [5]. The categorization task was performed by training an one-vs-all classifier for each of the subtasks. We only tuned the cost-factor parameter "-j" of SVMlight, which adjusts the weight of training error on positive training cases relative to that of negative cases. The performance of SVM classification with different cost-factors was evaluated by the built-in leave-one-out (LOO) estimation in SVMlight. In some preliminary experiments, we found that the performance of SVMlight on the training data set tends to reach a plateau as the cost-factor is increased, and we decided to set the cost-factor parameter to 20 for all our experiments.

3.2 Vocabulary-based Text Representation

The full text for task II was processed as follows: (1) SGML tags were stripped; (2) Tokens consisting of letters and numbers were extracted; (3) Words from a stop word list were removed; (4) Tokens were stemmed using a Porter stemmer. We further constructed a vocabulary consisting of 21,000 tokens that were "biologically informative" using a protein-related corpus consisting of MEDLINE abstracts and titles associated with Gene Ontology terms from the Gene Ontology Annotation [2] The tokens were extracted from the corpus and the mutual information of tokens with respect to GO terms was calculated, sorted and 21,000 tokens with high mutual information were retained.

For the SVM classification, a document is represented as a vector in a vector space. The element of the vector is further weighted using tf-idf weighting scheme and the length of the vector is normalized to 1 [7]. Such a vector representation using the original vocabularly was referred to as vocabulary-based text representation (VocRep).

3.3 Semantic Text Representation

The VocRep of the training text has a key drawback: the text is represented by a sparse vector in a very high dimensional space. Within such a space, data points tend to spread far apart and the SVM decision surface learned in such a space is likely to over-fit the training data, especially when a relatively small number of training cases are available. The ambiguities of natural language, i.e., polysemy and synonym, further complicate the situation because variations of word usages may easily cause two documents with similar semantic contents to be separated far away within the space.

To alleviate such a problem, we applied a probabilistic topic model, the Dirichlet Allocation Model (LDA) [1, 4] to extract semantic topics from the corpus and represent the text documents within this reduced semantic space. LDA treats a document as a mixture of words from different topics and applies probabilistic inference to extract the semantic topics from a corpus in an unsupervised way. We have applied the model on the combined training and test data sets and extracted 400 semantic topics from the corpus. The semantic topic for each word in the corpus was then inferred, which allowed us to represent a document as a vector within the semantic space defined by these 400 subtopics. Each element of the vector corresponds to the number of words within the document that belongs to the corresponding topic. Such a text representation is referred to as SemRep.

Semantic representation by the LDA has the following advantages: (1) Text is classified according to the semantic content; (2) Projecting text into a semantic space allows the documents that share few common words yet have similar semantic content to be closer in the vector space, thus increases the sensitivity of the classifier; (3) Dimension reduction increases the generality of the trained classifier; (4) LDA is computationally much less expensive than the classical latent semantic indexing (LSI).

Table 4: Comparison of VocRep and SemRep

Method		Task			
		A	E	G	T
F-score	VocRep	0.725	0.210	0.281	0.417
	SemRep	0.843	0.482	0.617	0.611
F-score	VocRep	0.363	0.152	0.159	0.283
	SemRep	0.317	0.154	0.160	0.268
Utility	VocRep	0.708	0.207	0.239	0.416
	SemRep	0.793	0.458	0.424	0.607

3.4 Semi-supervised learning

To further enhance the classification performance of the SVM, especially for the categories with few training cases, we augmented the positive training set using a semi-supervised learning approach described in [9]. This algorithm is based on the graph theory and allows the labels of training examples to be propagated to the unlabeled data. The newly-labeled pseudo-positive cases would then be incorporated into the training data set to enhance training of the SVM classifier. We constructed a weighted undirected graph, in which the vertices are the training documents and the weights of edges connecting the vertices are defined as:

$$w_{ij} = exp(\frac{1}{0.03}(1 - \frac{v_i' v_j}{|v_i| \times |v_j|}))$$

where w_{ij} is the weight of the edge connecting vertices i and j; $v_i' v_j$ is the dot product of the vectors for documents i and j. The term $|v_i|$ stands for the norm of the vector v_i. As negative examples, we have manually selected 500 training documents that do not belong to any of the four categories. These negative cases were used in combination with the provided positive training cases to perform the label-propagation experiments. For each category, we obtained 100 pseudo positive cases to augment the training cases for the category. Then, the SVM models were re-trained using the augmented data sets.

3.5 Result Analysis

We first compare VocRep and SemRep in terms of the training performance of SVM in Table 4 We see that SemRep significantly increases the utility and recall of SVMlight based on the leave-one-out evaluation on the training data, though it slightly decreases the F-score in some cases.

However, the improvement on the two relatively more difficult categories (i.e., E and G) is quite consistent for both F-score and utility, indicating that the LDA model can capture the major directions of the semantic structure of the corpus while maintaining the discriminative power in these cases.

We further examine the effect of semi-supervised learning in Table 5. We see that the semi-supervised learning approach significantly improves recall for the subtasks with very few positive training cases, namely the E and T subtasks, while the benefit on the other two subtasks is less significant.

Table 5: Recall of before and after semi-supervised learning

Method	Task			
	A	E	G	T
Before	0.840	0.531	0.617	0.639
After	0.867	0.691	0.630	0.861

In comparison with other groups' submissions, our official results are very good by F-score, but poor by utility. We believe that the main reason why the utility values of our runs are poor is because the SVM method we used appears to be have some ceiling for recall, probably because of the sparse training data. In the future, we plan to explore how to address this problem by de-regularizing SVM as well as other classifiers.

4 Summary

In summary, our ad hoc retrieval experiments are focused on studying how to optimize retrieval with semi-structured queries. The results show that exploiting the query structure is beneficial and the proposed theme-based language models are effective in performing both pseudo feedback and relevance feedback for such semi-structured queries. Our categorization experiments are focused on studying the effectiveness of using LDA for

dimension reduction. The experiment results demonstrate that LDA is capable of capturing meaningful semantic contents of text documents and such features are useful in improving SVM performance. We also found that the SVM approach appears to have some limitation in optimizing strongly biased utility functions and semi-supervised learning is beneficial to alleviate this problem.

References

[1] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] E. Camon, D. Barrell, V. Lee, E. Dimmer, and R. Apweiler. The gene ontology annotation (GOA) database–an integrated resource of go annotations to the uniprot knowledgebase. *Silico Biology*, 4, 2003.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statist. Soc. B*, 39:1–38, 1977.

[4] T. L. Griffiths and M. Steyvers. Finding scientific topics. In *Proc Natl Acad Sci USA*, volume 101 Suppl 1, pages 5228–35, 2004.

[5] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, 1998.

[6] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR'01*, pages 111–119, Sept 2001.

[7] D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[8] C. Zhai and J. Lafferty. Model-based feedback in the KL-divergence retrieval model. In *Tenth International Conference on Information and Knowledge Management (CIKM 2001)*, pages 403–410, 2001.

[9] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML 2003*, 2003.