Labeling Documents in Search Collection: Evolving Classifiers on a Semantically Relevant Label Space

Ramakrishna B Bairi
IITB-Monash Research Academy
IIT Bombay
Mumbai, India
bairi@cse.iitb.ac.in

Ganesh Ramakrishnan
Department of CSE
IIT Bombay
Mumbai, India
ganesh@cse.iitb.ac.in

ABSTRACT

Associating meaningful label or category information with every document in a search collection could help in improving retrieval effectiveness. However, identifying the right choice of category tags for organizing and representing a large digital library of documents is a challenging task. A completely automated approach to category creation from the underlying collection could be prone to noise. On the other hand, an absolutely manual approach to the creation of categories could be cumbersome and expensive. Through this work, we propose an intermediate solution, in which, a global, collaboratively-developed Knowledge Graph of categories can be adapted to a local document categorization problem over the search collection effectively. We model our classification problem as that of inferring structured labels in an Associative Markov Network meta-model over SVMs, where the label space is derived from a large global category graph.

Keywords

Large scale text classification, Text categorization, Topic identification, Multi-label classification, Personalization

1. INTRODUCTION

With the growth of digital data in the form of news, blogs, web pages, scientific articles, books, images, sound, video, social networks and so on, the need for effective categorization systems to organize, search and extract information becomes self-evident.

Categories associated with a document can act as semantic summaries which can help in catering to the user intention behind a query [3], diversifying the search results [1], semantic search [7], personalized search [12], grouping search results for effective presentation and/or access [5] and better weighting terms in IR [9], to name a few.

An important aspect in building a categorization system is the choice of categories. Categories that are very generic, such as News, Entertainment, Technical, Politics, Sports, and the like may not be useful. Thousands of articles could accumulate under each such category and searching for the required piece of information could still be a challenge. For semantic search to be useful, it can be expected that the category space covers a reasonably wide range of queries that might be expected. Creation of fine-grained categories and assigning them to documents needs domain experts and is a laborious task.

Adopting predefined categories from an existing classification system (such as Reuters text classification dataset) may not be always suitable. Such a strategy could lead to (i) under or over specific categories (ii) failure to capture user intention (iii) failure to evolve with time.

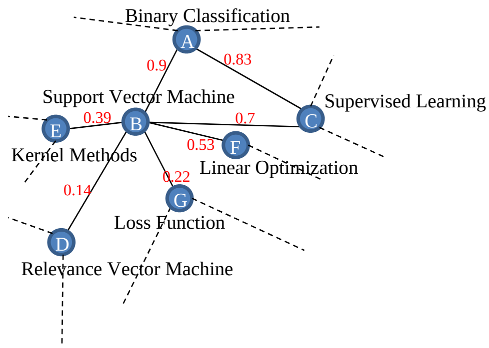


Figure 1: A part of the Knowledge Graph

In this paper we present our attempts to address these practical issues in designing a document categorization system to semantically cover a given document collection. Throughout this paper, we refer to our system as EVO. We assume as input to our system, an extremely large Knowledge Graph (KnG) whose nodes are categories, and edges are relationship between the categories. Each category is accompanied by some description of that category. Every edge is also associated with a score between 0.0 to 1.0 indicating the strength of the relationship. This score can be generated using document similarity measurement techniques (such as Jaccard, Cosine, Kernels or semantic similarity methods). Such a knowledge graph can be built collaboratively. For experimental purposes we treat Wikipedia as a knowledge graph. Wikipedia's 4M articles cover the terminology of nearly any document collection [15], which could make it a good candidate for KnG. A part of KnG is shown in Figure 1. Next, we need sound techniques for adopting the categories in this KnG to a given collection of documents. In this paper, we propose a technique to solve this problem by learning a model to project the documents into a localized subset of the categories in KnG; this is done by capturing various signals from the documents, exploiting the knowledge in KnG and evolving the category specific classifiers (say SVMs) via user feedback.

2. FORMAL PROBLEM STATEMENT AND SOLUTION PROPOSAL

We assume that a knowledge graph exists with *very large* collection of possible categories (that can cover the terminology of nearly any document collection; for example, Wikipedia) $C = \{C_i\}_{i=1}^{i=f}$ as nodes, and the relationship between them as edges. There is a description associated with each category and each edge between selected pairs of categories is associated with a score reflecting the strength of the relationship between the two categories. We further assume that an organization receives documents in batches $D_1, D_2, ...$ where each batch D_j is received at j^{th} time period (say, j^{th} week/month and the like.) The organization needs to adopt (subset) the categories in KnG to logically build an organization-specific category catalog $C^{org} \subseteq C$ and at the same time, evolve some models to classify all $d_i \in D_j$ into C^{org}. More specifically, we assume the following goals:

1. Learning a personalised model for the association of the categories in KnG to a document collection through knowledge propagation and feature design
2. Building an evolving multi-label categorization system to categorize documents into C^{org}.

The eventual goal is to accurately identify suitable categories $\{C_{i_1}, ...C_{i_T}\}$ for every input document $d_i \in D_j \; \forall i, j$. If one could learn an SVM classier for every category in the KnG, identifying all suitable categories for a document would entail determining which classifiers label the document as positive. However, learning such classifiers upfront is prohibitively expensive because, the KnG is usually very large (for example, Wikipedia has four million titles) making it impractical to learn a classifier (SVM) for every category in KnG using limited training data. Hence, it is a challenging task to develop a classification system which can identify a subset of the millions of categories that suit an organization. We attempt to solve this problem from a new perspective of knowledge propagation techniques, where the categories in KnG exchange the knowledge of similarity to the documents and the classifier scores (when available) to draw a collective inference on the categories relevant to the document. We explain our technique in detail throughout the rest of this paper. Figure 3 illustrates the overall process of evolving a personalized classifier.

It has been observed that a document that is tagged with a category is expected to contain features such as keywords/phrases that are indicative of that category [13, 8]. For example, the text shown in Figure 2, contains several words/phrases that are indicative of some of the category titles in the KnG (Wikipedia, in our examples.) Techniques such as [14, 8, 13] can be used for spotting such keywords/phrases. We refer to such categories as *candidate categories.* "Keywords Spotter" component in Figure 3 detects candidate categories. However, some of these categories could be either (a) misleading or (b) not relevant in determining the "interesting categories." As an illustration of (a), consider, in Figure 2, the category "Jikes RVM" (which is picked up due to the spotted keyword RVM,) which means Java JVM—not relevant to the document. Thus, the word "RVM" is misleading as a feature. On the other hand, while the category "Cancer" is relevant to the document, the user may want to restrict the choice of categories to the computer science domain, and may therefore, not be interested in categories like "Cancer," thus making a case for (b). Our goal is to develop a personalized categorization system that has the capacity to evolve and learn how to accurately identify only relevant categories. This can be achieved by incrementally learning a classier for each class, based on user feedback. We expect the classifier training to result in feature weights such that the effect of misleading and irrelevant features described above is minimized.

Another benefit of having a candidate categories identification phase is that, it allows us to evolve C^{org} with more categories when the documents with new categories are seen by our system. The spotter can recognize these new categories which can become part of C^{org} eventually. By this process, we overcome the problem of under-specified categories that prevails in the classification systems with predefined categories. However, in the process, we may result in over-specified categories, if we do not control the addition of new categories to C^{org}. We observed that, simple heuristics such as generating a histogram of categories with the number of documents classified under them and then pruning the categories that have very few or very high number of documents can work reasonably well in practice. In addition, our user feedback mechanism, which we explain later in the paper, will also help in limiting the number of categories in C^{org}. More sophisticated approaches to address under or over specified categories using category hierarchies from KnG, which we are exploring currently, will form the part of our future work.

We also observe that categories that get assigned to a document either exhibit semantic relations such as "associations,"[1] "descriptions overlap," and the like or tend to be frequently assigned together (that is, tend to co-occur) in a particular instance of the classification exercise. For example, with the Reuters RCV1-v2 dataset, we observe that all pairs of categories that co-occur even once in the training set, co-occur multiple times in test set. In other instances of classified data such as DMOZ or the Yahoo! Directory, we make an additional observation that co-occurring categories exhibit semantic relations such as "association." For example, the category "Linear Classifier" is related to categories such as "Kernel Methods in Classifiers," "Machine Learning," and the like, and are observed to co-occur as labels for a document on "Classifiers." Another illustration: categories "Support Vector Machines" and "Kernel Methods" exhibit a lot of overlap in their textual descriptions. To sum up, we identify two types of informative features to identify relevant categories for each document: (i) a feature that is a function of the document and a category, such as the category-specific classifier scoring function evaluated on a document and (ii) a feature that is a function of two categories, such as their co-occurrence frequency or textual overlap between their descriptions. We find Associative Markov Network (AMN) [17], a very natural way of modeling these two types of features. Next, we provide a more detailed description of our modeling of this problem as an Associative Markov Network.

For every input document d, we construct a Markov Network (MN) from the candidate categories, such that, each node represents a candidate category $C_i \in C$ and edges represent the association between the categories. Modeling inter-category relations through edges serves two impor-

[1] http://marciazeng.slis.kent.edu/Z3919/44association.htm

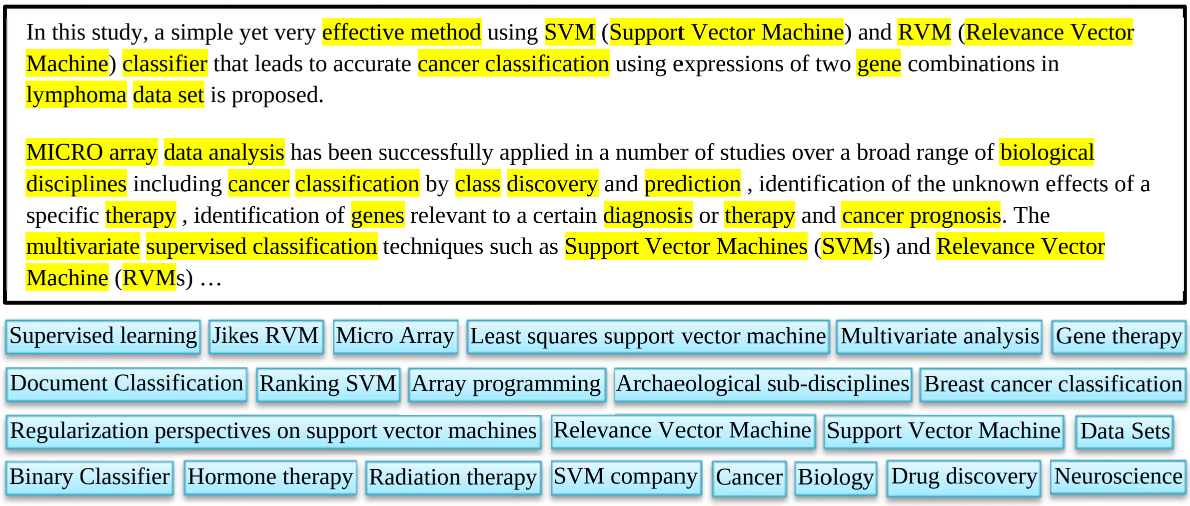


Figure 2: Document with detected keywords (in yellow) and sample candidate categories (in blue)

tant purposes in our approach: i) When a new organization starts categorizing documents, the classifier models are initially not tuned. The only information available to the categorization system are the category descriptions. It is not practical to assume that perfect descriptions will be available for every category. In such cases, the relationship between the categories can help propagate descriptions across categories via their neighbors. ii) As part of learning the model parameters, the system solicits user feedback on some of the suggested categories for a document. Based on the feedback, the category-specific model (SVM) is updated. The category relationship helps in propagating the learning to the neighbors. This reduces the number of feedbacks needed to learn the model parameters. We will illustrate both these advantages in our experimental section.

Our aim is to learn to assign a binary label (0/1) for every category node C_i in the above MN. Label 1 indicates that the category C_i is valid for the document d and 0 indicates invalid. The collective assignment of labels for all the nodes in the Markov network produces relevant categories for the document d. As we see later in the paper, optimal assignment of these labels can be achieved through MAP inference using Integer Linear Programming.

The "Amn + SVM classifier" component in Figure 3 performs the AMN inference using the learned model parameters and user feedback (along with user defined constrains, explained later in this paper.)

The "Classifier Trainer" component in Figure 3 helps in training the category specific classifiers and updates model parameters.

3. LEARNING PERSONALIZED CLASSIFIER

3.1 Building AMN model from categories

For a given document d, we create an MN $G = (N, E)$, whose nodes N are the candidate categories from the KnG and edges E are the association between them, as present in KnG.

In an AMN, only node and edge potentials are considered. For an AMN with a set of nodes N and edges E, the conditional probability of label assignment to nodes is given by

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod \varphi\left(\mathbf{x}_i, y_i\right) \prod \psi\left(\mathbf{x}_{ij}, y_i, y_j\right) \qquad (1)$$

We use notation $\mathbf{x}_i$ to denote a set of node features for the candidate category node C_i and $\mathbf{x}_{ij}$ to denote the set of edge features for the edge connecting C_i and C_j. y_i and y_j are the binary labels for nodes C_i and C_j.

The node features in AMN determine the relevance of a category to the input document d and the edge features capture the strength of the various associations between the categories. Note, here the node features $\mathbf{x}_i$ are computed by considering the node description and the input document text. Hence the above distribution is for a given document d.

Z denotes the partition function given by $Z = \sum_{\mathbf{y}'} \prod \varphi\left(\mathbf{x}_i, y'_i\right) \prod \psi\left(\mathbf{x}_{ij}, y'_i, y'_j\right)$.

A simple way to define the potentials φ and ψ is the log-linear model. In this model, a weight vector is introduced for each class label $k = 1..K$. The node potential φ is then defined as $log\varphi\left(\mathbf{x}_i, y_i\right) = \mathbf{w}_n^k \cdot \mathbf{x}_i$ where $k = y_i$. Accordingly, the edge potentials are defined as $log\psi\left(\mathbf{x}_{ij}, y_i, y_j\right) = \mathbf{w}_e^{k,l} \cdot \mathbf{x}_{ij}$ where $k = y_i$ and $l = y_j$. Note that there are different weight vectors $\mathbf{w}_n^k \in \mathbb{R}^{d_n}$ and $\mathbf{w}_e^{k,l} \in \mathbb{R}^{d_e}$ for the nodes and edges.

Using the indicator variables y_i^k we can express the potentials as: $log\varphi\left(\mathbf{x}_i, y_i\right) = \sum_{k=1}^K \left(\mathbf{w}_n^k \cdot \mathbf{x}_i\right) y_i^k$ and $log\psi\left(\mathbf{x}_{i,j}, y_i, y_j\right) = \sum_{k=1}^K \left(\mathbf{w}_e^{k,l} \cdot \mathbf{x}_{ij}\right) y_i^k y_j^l$; where y_i^k is an indicator variable which is 1 if node C_i has label k and 0, otherwise.

To bring in the notion of association, we introduce the constraints $\mathbf{w}_e^{k,l} = 0$ for $k \neq l$ and $\mathbf{w}_e^{k,k} \geq 0$. This results in $\psi\left(\mathbf{x}_{ij}, k, l\right) = 1$ for $k \neq l$ and $\psi\left(\mathbf{x}_{ij}, k, k\right) \geq 1$. The idea here is that edges between nodes with different labels should be penalized over edges between equally labeled nodes.

Learning feature weight vectors is based on Max Margin training, which is of the form

$$\begin{aligned} &\underset{\mathbf{w},c}{\text{argmin}}\ \frac{1}{2}\left\|\mathbf{w}\right\|^2 + c\xi \\ &\quad s.t.\ \mathbf{w}\mathbf{X}\hat{\mathbf{y}} + \xi \geq \max_{\mathbf{y}} \mathbf{w}\mathbf{X}\mathbf{y} + (|N| - \mathbf{y}.\hat{\mathbf{y}}_\mathbf{n});\ \ w_e \geq 0 \end{aligned}$$

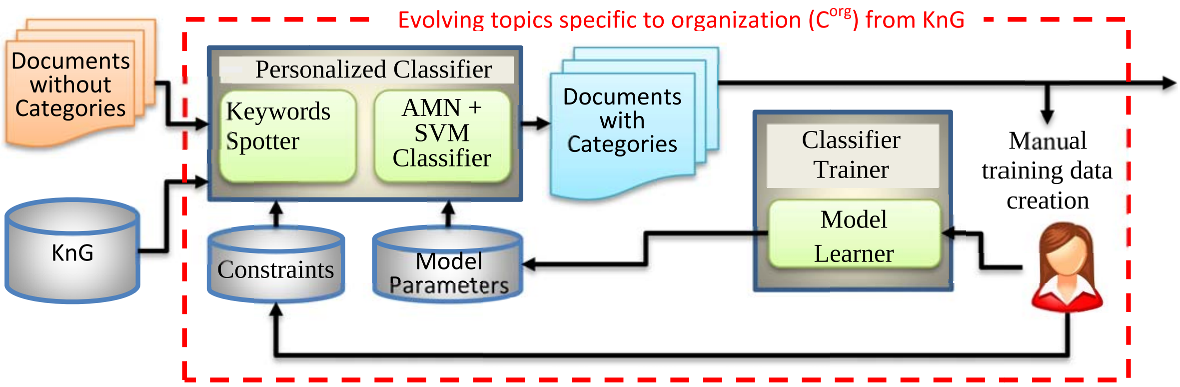


Figure 3: Architecture of KnG category Personalization

Using compact representation, we define the node and edge feature weight vectors $\mathbf{w}_n = \left(\mathbf{w}_n^1, ..., \mathbf{w}_n^K\right)$ and $\mathbf{w}_e = \left(\mathbf{w}_e^{1,1}, ..., \mathbf{w}_e^{K,K}\right)$, and let $\mathbf{w} = (\mathbf{w}_n, \mathbf{w}_e)$ be the vector of all the weights. Also, we define the node and edge labels vectors, $\mathbf{y}_n = (..., y_i^1, ..., y_i^K, ...)^{\mathrm{T}}$ and $\mathbf{y}_e = (..., y_{ij}^{1,1}, ..., y_{ij}^{K,K}, ...)^{\mathrm{T}}$, where $y_{ij}^{k,l} = y_i^k y_j^l$, and the vector of all labels $\mathbf{y} = (\mathbf{y}_n, \mathbf{y}_e)$. The matrix $\mathbf{X}$ contains the node feature vectors $\mathbf{x}_i$ and edge feature vectors $\mathbf{x}_{ij}$ repeated multiple times (for each label k or label pair k, l respectively), and padded with zeros appropriately. $\widehat{\mathbf{y}}$ is the vector of true label assignments given by the training instance. $|N|$ is the number of nodes in the graph G.

We request the reader to refer to[17] for details of solving this optimization.

3.2 Inferring categories for a document

The problem of inference is to select a subset of nodes (that is, categories) from G that have the highest probability of being relevant to the input document. To model this selection, we attach a binary label $\{0, 1\}$ to a node. A node C_i with label 1 is considered to be a valid category for the input document and invalid if its label is 0.

Correctly determining the categories for the input document is equivalent to solving the MAP optimization problem in (2).

$$\max_{\mathbf{y}} \sum_{i=1}^{N} \sum_{k=0}^{1} \left(w_n^k \cdot x_i\right) y_i^k + \sum_{(ij)\in E} \sum_{k=0}^{1} \left(w_e^k \cdot x_{ij}\right) y_{ij}^k (2)$$
$$s.t. \quad y_i^k \geq 0, \quad \forall i, k \in \{0, 1\};$$
$$\textstyle\sum_{k=0}^{1} y_i^k = 1, \quad \forall i$$
$$y_{ij}^k \leq y_i^k, \quad y_{ij}^k \leq y_j^k, \quad \forall ij \in E, k \in \{0, 1\}$$
$$y_i^0 \quad = 1 \ \ \forall i \text{ with Hard Constraints}$$

The variables y_{ij}^k represent the labels of two nodes connected by an edge. The inequality conditions on the fourth line are a linearization of the constraint $y_{ij}^k = y_i^k \wedge y_j^k$; We explain Hard Constraints in section 3.4.

The above MAP inference produces the optimum assignment of labels y_i^k that maximizes the probability function in Equation 1. It can be shown that the Equation 2 produces integer solution when unique solution exists. When $y_i^1 = 1$, we attach the label 1 to the node C_i, and when $y_i^0 = 1$,we attach the label 0 to the node C_i. (Note, both y_i^0 and y_i^1 cannot be 0 or 1 simultaneously, due the second constraint.)

3.3 Defining AMN Node and Edge features

3.3.1 *Node Features for knowledge propagation*

We divide the node features $\mathbf{x}_i$ into two types : i) Global node features $\mathbf{x}_i^s$ and ii) Local node features SVM_i^0 and SVM_i^1. The node feature vector becomes $\mathbf{x}_i = \left[\mathbf{x}_i^s; SVM_i^0; SVM_i^1\right]$.

Global features: These features aid in capturing the structural similarity of a node to the input document through a combination of different kernels such as Bag of Words kernels, N-gram kernels, Relational kernels, among others. The values of global features do not change over time. An example of global feature could be, cosine similarity between the bag of words representations of a document and the description associated with a node in the KnG.

Local features: These features aid in the personalization of KnG. Essentially, we learn an SVM model for a category based on user feedback. It is very much possible to consider the use of other machine learning models such as decision trees, logistic regression, *etc.* Our choice of SVM was based partly on the fact that all our baselines employ SVMs and partly on the fact that SVMs are known to yield high accuracies. We employ the decision function of the classifier as a node feature in the AMN. The SVM decision function for each category node takes as input, the document d (as a TF-IDF vector), and evaluates $Svm_{C_i}(d) = \mathbf{w}_{C_i}^T \mathbf{d} + b_{C_i}$, where $\mathbf{w}_{C_i}$ and b_{C_i} are the SVM parameters learnt for the category C_i. The output of the SVM decision function is positive if C_i is relevant for the document d and negative if not relevant. We also treat the output of decision function to be 0 if the SVM model is not available for the category C_i. We introduce two features in the node feature vector $\mathbf{x}_i$, *viz*, SVM^1 and SVM^0, denoted using the notation SVM_i^1 and SVM_i^0.

The feature value is computed as follows:

$$SVM_i^1 = \begin{cases} \gamma_i Svm_{C_i}(d) & if \ \ Svm_{C_i}(d) \geq 0 \\ 0 & Otherwise \end{cases}$$

$$SVM_i^0 = \begin{cases} -\gamma_i Svm_{C_i}(d) & if \ \ Svm_{C_i}(d) < 0 \\ 0 & Otherwise \end{cases}$$

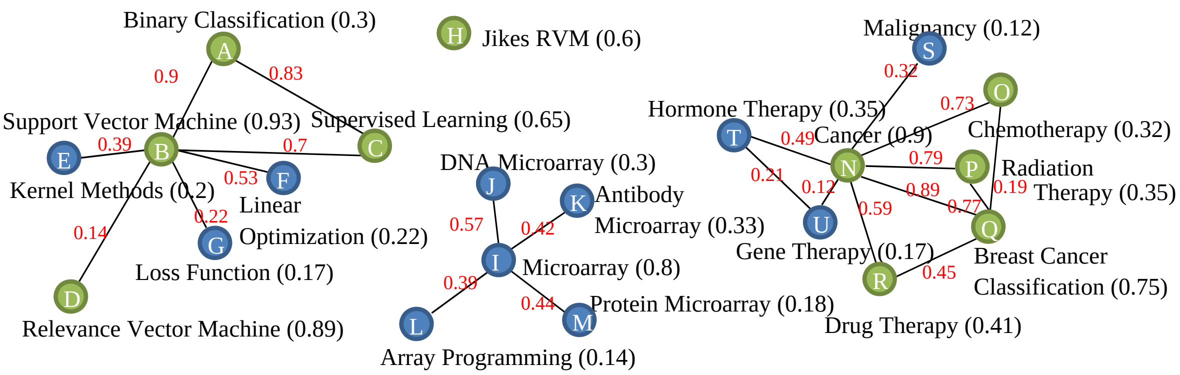


Figure 4: Knowledge Propagation: 1. Nodes B and C with label 1 force the strongly associated neighbor node A to assume label 1. We say that, knowledge from node B and C propagates to node A. 2. Though node I seems to be valid for the document (with high node potential), given the context, it is not. Strongly associated neighbors of I, that is, nodes J,K,L,M which have low node potentials force the node N to attain label 0. Here again we say that, knowledge flows from J,K,L,M to I.

γ_i is the *damping factor*, which reflects the confidence in SVM classifier for the category C_i. When the classification system is initially deployed, we believe that the SVM models would not have been sufficiently trained. Hence, the SVM feature values might not initially provide reliable signals in deciding the relevance of a category to the input document. As the categorization system matures by training via user feedback, the SVM models get trained so that we can increasingly start trusting the SVM scores. We can control this through the *damping factor* γ_i, that increases for a category with the number of feedbacks received for that category. We define γ_i to be:

$$\gamma_i = \begin{cases} 0 & \text{if } confidence\left(Svm_{C_i}\right) < \mathcal{T} \\ confidence\left(Svm_{C_i}\right) & \text{otherwise} \end{cases},$$

where $\mathcal{T}$ is the user defined threshold, $confidence\left(Svm_{C_i}\right)$ is the $\xi\alpha - estimator$ of F1 score of Svm_{C_i} computed as in [10]. These estimators are developed based on the idea of leave-one-out estimation of error rate. However, leave-one-out or cross validation estimation is a computationally intensive process. On the other hand, $\xi\alpha - estimator$ [10] can be computed at essentially no extra cost immediately after training every Svm_{C_i} using the slack variables ξ_i in the SVM primal objective and α_i Lagrange variables in the SVM dual formulation.

Due to the associative property of AMN, the SVM parameters learned for a node can also influence the label assignments of its neighbors. In other words, if there is a strong edge potential between categories C_i and C_j, the SVM score propagates from C_i to C_j. This helps in correct detection of the label of node C_j even though there may not be a trained SVM classifier available for node C_j. The example in Figure 4 illustrates the knowledge propagation between highly associated (that is, with high edge potential) nodes. This is precisely what we aim to model using an AMN.

3.3.2 Edge features for knowledge propagation

Edges between categories in a Markov Network represent some kind of association between them. An Edge feature vector $(\mathbf{x}_{ij})$ contains feature values that encourage the categories C_i and C_j connected by the edge to have same label if there is a strong relationship between them. The strength of relationship is discovered through combinations of multiple Kernels. Let $K_1\left(C_i, C_j\right) \cdots K_M\left(C_i, C_j\right)$ be M Kernels that measure the similarity between C_i and C_j. Example Kernels include Bag-of-Words Kernel, Bi-gram Kernel, Trigram Kernel, Relational Kernel, *etc.* Further we assume (without loss of generality) that these Kernels return normalized values (between 0 and 1). We define the feature vector $\mathbf{x}_{ij}$ to have M features that signal $y_i = 1$, $y_j = 1$ and M features that signal $y_i = 0$, $y_j = 0$. The feature vector $\mathbf{x}_{ij}$ is defined as follows $\forall 1 \le m \le M$

$$\mathbf{x}_{ij}\left[m\right] = K_m\left(C_i, C_j\right) \times \left(\log\varphi\left(\mathbf{x}_i, 1\right) + \log\varphi\left(\mathbf{x}_j, 1\right)\right)$$

$$\mathbf{x}_{ij}\left[M+m\right] = K_m\left(C_i, C_j\right) \times \left(\log\varphi\left(\mathbf{x}_i, 0\right) + \log\varphi\left(\mathbf{x}_j, 0\right)\right)$$

Note that $\log\varphi\left(\mathbf{x}_i, 1\right)$ is the node potential of C_i when it is labeled 1 and $\log\varphi\left(\mathbf{x}_i, 0\right)$ is the node potential when it is labeled 0. Essentially, when the similarity between nodes C_i and C_j is high, these features collectively favour similar labels on both the nodes C_i and C_j.

3.4 Enforcing category Constraints

In the process of personalizing the KnG, users can indicate (via feedback) that a category C_i suggested by the system should never reappear in future categorization, because the organization is not interested in that category. For example, an organization working in the core area of Computer Science may not be interested in a detailed categorization of cancers, even though there may be some documents on classification algorithms for different types of cancers. The system remembers this feedback as a *hard constraint*. By *hard constraint* for a category C_i, we mean the inference that is subject to a constraint set that includes $y_i^0 = 1$, as in Equation 2. If categories C_i and C_j are *related*, we would expect the effect of this constraint to propogate from C_i to C_j and encourage y_j^0 also to become 1. As shown in the example in Figure 5, if the user suppresses the category *Cancer* by introducing a hard constraint, the AMN inference will try to suppress *related* categories as well. This is precisely what we aim to model using an AMN.

3.5 Inferring C^{org}

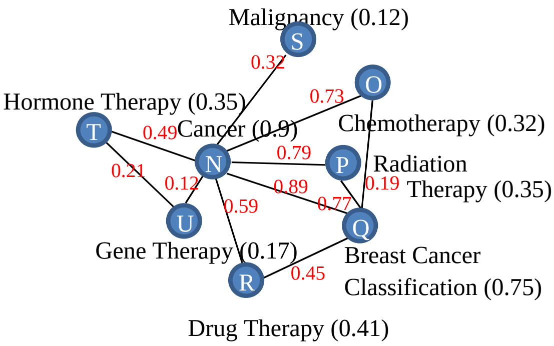


Figure 5: Constraint Propagation: By applying a "never again" constraint on node N, the label of Node N is forced to 0. This forces labels of strongly associated neighbors (O,P,Q,R) to 0. This is due to the AMN MAP inference, which attains maximum value when the labels of these neighbors (with high edge potentials) are assigned label 0.

So far, we have shown how to infer a set of categories for a document d. We have indicated that these categories come from C^{org}. Essentially, $C^{org} \subseteq C$ is hidden behind our model parameters (AMN and SVM) and hard constraints, which keeps updating with every feedback. For any new document d, when we apply our inference logic, we essentially derive the categories for d from C^{org}. However, if all the members of C^{org} need to be enumerated, we need to infer all the categories of all the documents seen by the organization so far, with the current set of model parameters and hard constraints. However, in practice, we may not have to enumerate C^{org} for the functioning of our system.

Evolving C^{org} over time has two dimensions: (i) evolving C^{org} when new documents with new categories (which exist in KnG) are seen by our system, and (ii) evolving C^{org} when new categories are added to KnG. For the first case, assuming that the collaboratively built knowledge graph KnG is up-to-date with all the categories, our spotting phase identifies the features in the document corresponding to the new categories and adds them to the *candidate categories*. If these categories get label 1 during the inference, they are considered to be part of C^{org}. For the second case, the challenge lies in updating the already classified documents with the new categories added to KnG. One strategy of handling this could be to look at the neighborhood of newly added categories in KnG, retrieve the already classified documents that have categories present in this neighborhood and reassign categories to these documents by repeating our inference algorithm. In our current work, we limit the evolution C^{org} to case (i). Handling of case (ii) will be part of our future work.

3.6 A note on user feedback and training per category classifiers

Well trained category specific classifiers (SVMs in our case) can boost the accuracy of classifiers. Note that, it is not required to train the classifiers attached to every category. Whenever available, the knowledge of classifier's decision propagates to the neighboring nodes in KnG. This is the whole point setting up AMN inference over KnG. This minimizes the number of classifiers to be trained. However, this training can lead to significant cognitive load on the users. To reduce this cognitive load and to achieve a better learning rate, we can adopt the Active Learning strategy, where we seek feedback from the user on select categories for select documents. Development of effective Active Learning strategy will be part of our future work.

4. EXPERIMENTS AND EVALUATION

4.1 Global Knowledge Graph (KnG)

We extract Wikipedia Page/Article titles and add them to our KnG. We also construct description text for each category in KnG from the first few paragraphs (gloss) of Wikipedia's page. We introduced edges between the nodes connected via hyperlinks to capture the association in terms of text overlap, title overlap, gloss overlap and anchor text overlap.

4.2 Data-sets

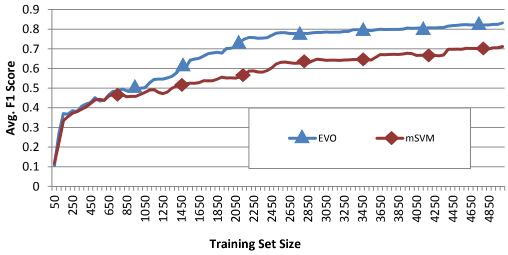
We report experiments on the RCV1-v2 benchmark dataset. Our choice of datasets was based on the existence of at least 100 class labels in the dataset. The Reuters RCV1-v2 collection consists of 642 categories and a collection of 23,149 documents in the training set and 781,265 documents in the test set.

4.3 Evaluation Methodology

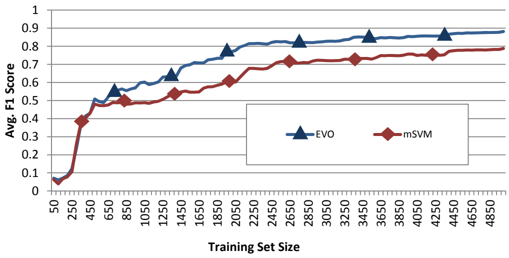
In these experiments we demonstrate how, on a standard classification dataset, the Markov network helps propagate learnings from a category to other *related* categories. While the AMN model exploits inter-class correlation, the per-class SVM model incorporates feedback on document labels more directly. In the absence of inter-class correlation, our model degenerates to a multiclass SVM classifier. To demonstrate this, we report experiments with two different strategies for sampling documents: (i) clustered sampling and (ii) random sampling.

Experiments on correlated categories (Clustered Sampling).
In this setting, we selected 66 pairs of *related* Reuters categories, spanning 96 categories. For *e.g.*, the categories MANAGEMENT and MANAGEMENT MOVES are *related*. So are LABOR and LABOR ISSUES. Two clategories were considered *related* if the number of training documents carrying both labels, exceeded certain threshold. Our clustered sampling entailed sampling of documents that were labeled with both categories in any of the 66 pairs. We picked 5000 training documents and 2000 test documents using this clustered sampling procedure. We further divided the training set into 100 batches of 50 documents each. We iterated through the batches and in the k^{th} iteration, we trained our model (SVMs, γ_i, AMN feature weights) using training documents from all batches upto the k^{th} batch. For each iteration, we performed AMN inference on the sample of 2000 test documents. In Figure 6, we report the average F1 score on the test sample for each of 100 iterations.

The curves labeled EVO correspond to the F1 score of our system whereas the curves labeled mSVM are for the F1 score of a plain multiclass SVM. These graphs are plotted for experiments with different choices (0.1, 1, 10, 100; shown only for 1 and 10 here) of the SVM hyper-parameter C. As we expect, after around 20 iterations, some of the better-trained SVMs start propagating their learning to their cor-

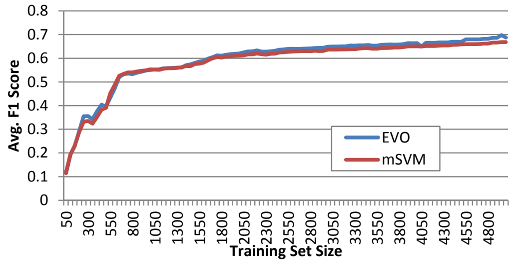


(a) With SVM parameter C=1

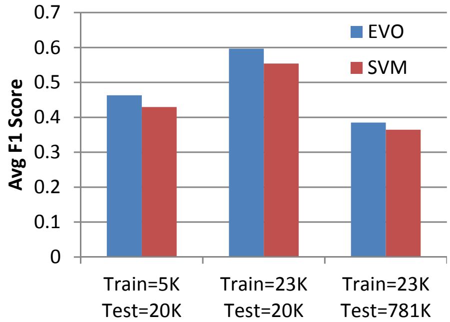


(b) With SVM parameter C=10

Figure 6: Comparison of avg (macro) F1 scores of our system (EVO) with SVM on different c values



(a) Increasing Training Set Size



(b) Aggregate

Figure 7: Comparison of avg (macro) F1 scores of our system (EVO) with SVM (random sampling)

related neighbors (enforced by AMN), hence boosting the overall F1 score. In other words, we can say that the learning rate in our model is faster than that of a plain SVM model. Hence, with a fewer training examples we can achieve the same level of accuracy/recall as that of a plain multiclass SVM trained with significantly more examples.

Experiments on uncorrelated/loosely correlated categories (Random Sampling).

When there is no correlation or very little correlation between the categories, the AMN will not contribute much to the inference. To study this case, we selected all the Reuters categories and randomly sampled 2000 test documents from the Reuters standard test split and 5000 training documents from the training split. Rest of the evaluation procedure is same as in the previous case.

Figure 7a shows the avg F1 score over 100 iteration. Since there is not much correlation between the categories, in most of the iterations, F1 score of our model (EVO) follows the F1 score of multiclass SVM (mSVM). Due to the presence of small number of correlated categories in the test set, we see a small increase in the F1 score after about 40 iterations.

Figure 7b depicts the avg F1 scores of EVO and mSVM over the entire Reuters collection of test and train documents. EVO performs about $2-5\%$ better.

5. PRIOR WORK

Text classification/categorization is a well studied area in machine learning under a variety of settings such as supervised, unsupervised, and semi-supervised.

[16] present an algorithm to build a hierarchical classification system with predefined class hierarchy. Their classification model is a variant of the Maximum Margin Markov Network framework, where the classification hierarchy is represented as a Markov tree.

Topic Modeling in an unsupervised setting has been studied in CTM[4], PAM[11], NMF[2], which identify topics as a group of prominent words. Discovering several hundred topics using these techniques turns out be practically challenging with a moderately sized system. In addition, finding a good representative and grammatically correct topic name for a group needs additional effort.

Nadia and Andrew [6] explore multi-label conditional random field (CRF) classification models that directly parameterize label co-occurrences in multi-label classification. They show that such models outperform their single label counterparts on standard text corpora. We draw inspiration from [6] and jointly make use of relations between the categories in KnG along with the category similarity to the document to learn the categories relevant to a document.

Medelyan [15] detect topics for a document using Wikipedia article names as category vocabulary. However, their system does not adapt to the user perspective. Whereas, our proposed techniques support personalized category detection.

6. CONCLUSION

We presented an approach for evolving an organization-specific multi-label document categorization system by adapting the categories in a global Knowledge Graph to a local document collection. It not only fits the documents in the digital library, but also caters to the perceptions of users in the organization. We address this by learning an organization-specific document categorization meta-model using Associative Markov Networks over SVM by blending (a) global features that exploit the structural similarities between the categories in the global category catalog and input document and (b) local features including machine learned discriminative SVM models in an AMN setup along with user defined constraints that help in localization of the global category catalog (Knowledge Graph). In the process, we also curate the training data. Currently our system works only with a flat category structure. We believe that our technique can be improved to handle a hierarchical category structure, which will form part of our future work.

Acknowledgements We thank Prof. Mark Carman from Monash University for this valuable comments and suggestions.

7. REFERENCES

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, WSDM '09, pages 5–14, New York, NY, USA, 2009. ACM.

[2] S. Arora, R. Ge, and A. Moitra. Learning topic models - going beyond svd. *CoRR*, abs/1204.1956, 2012.

[3] R. Bairi, A. A, and G. Ramakrishnan. Learning to generate diversified query interpretations using biconvex optimization. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 733–739, Nagoya, Japan, October 2013. Asian Federation of Natural Language Processing.

[4] D. Blei and J. Lafferty. Correlated Topic Models. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, 18:147, 2006.

[5] H. Chen and S. Dumais. Bringing order to the web: Automatically categorizing search results. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00, pages 145–152, New York, NY, USA, 2000. ACM.

[6] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 195–200, New York, NY, USA, 2005. ACM.

[7] R. Guha, R. McCool, and E. Miller. Semantic search. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 700–709, New York, NY, USA, 2003. ACM.

[8] K. M. Hammouda, D. N. Matute, and M. S. Kamel. Corephrase: Keyphrase extraction for document clustering. In *Proceedings of the 4th International Conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM'05, pages 265–274, Berlin, Heidelberg, 2005. Springer-Verlag.

[9] R. Jin, J. Y. Chai, and L. Si. Learn to weight terms in information retrieval using category information. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 353–360, New York, NY, USA, 2005. ACM.

[10] T. Joachims. Estimating the generalization performance of an svm efficiently. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 431–438, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[11] W. Li and A. McCallum. Pachinko allocation: DAG-structured mixture models of topic correlations. *Proceedings of the 23rd international conference on Machine learning*, pages 577–584, 2006.

[12] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *Knowledge and Data Engineering, IEEE Transactions on*, 16(1):28–40, Jan 2004.

[13] Z. Liu, W. Huang, Y. Zheng, and M. Sun. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 366–376, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[14] O. Medelyan and I. H. Witten. Thesaurus based automatic keyphrase indexing. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '06, pages 296–297, New York, NY, USA, 2006. ACM.

[15] O. Medelyan, I. H. Witten, and D. Milne. Topic indexing with wikipedia, 2008.

[16] J. Rousu, C. Saunders, S. SzedmÃ¡k, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, 7:1601–1626, 2006.

[17] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 102–, New York, NY, USA, 2004. ACM.