A Structure-sensitive Framework for Text Categorization

Ganesh Ramakrishnan*
IBM India Research Lab
New Delhi, India
ganramkr@in.ibm.com

Deepa Paranjpe†
IBM India Research Lab
New Delhi, India
dparanjp@in.ibm.com

Byron Dom
Yahoo! Inc.
Sunnyvale, CA
bdom@yahoo-inc.com

ABSTRACT

This paper presents a framework called Structure Sensitive CATegorization(SSCAT), that exploits document structure for improved categorizaton. There are two parts to this framework, *viz.* (1) Documents often have layout structure, such that logically coherent text is grouped together into fields using some mark-up language. We present a log-linear model, which associates one or more features with each field. Weights associated with the field features are learnt from training data and these weights quantify the importance of the field features in determining the category for the document. (2) We present a method that associates weights with words in phrasal constructs and exploits natural language structure of these constructs to boost weights of important words called *focus words*. These weights are learnt from example instances of phrasal constructs, marked with the corresponding *focus words*. The learning is accomplished by training a classifier that uses linguistic features obtained from the text's parse structure. The weighted words, in fields with phrasal constructs, are used in obtaining features for the corresponding fields. SSCAT was tested on the supervised categorization task of over one million products from Yahoo!'s online shopping data. With an accuracy of over 90%, our classifier outperforms Naive Bayes and Support Vector Machines. This not only shows the effectiveness of SSCAT but also strengthens our belief that linguistic features based on natural language structure can improve tasks such as text categorization.

Categories and Subject Descriptors

H.3.0 [**Information Storage and Retrieval**]: Indexing

General Terms

Framework, Algorithms, Experiments

*Work done while interning at Yahoo!, Bangalore.

†Contact Author.Work done while interning at Yahoo!, Bangalore.

Keywords

Text Categorization, Log-linear Model, Naive Bayes, Conditional Random Fields, Natural Language Processing

1. INTRODUCTION

Categorization is defined as the task of assigning a category, chosen from a pre-defined set or hierarchy of categories, to an instance under consideration. Most state-of-the-art categorizers use a supervised learning approach; a classifier is trained on manually categorized text documents. The feature space in most of these is obtained using simple word counts in the documents and this is typically referred to as a *bag of words* representation for the documents. However, the documents to be categorized, often come with some "semi-structure", rendered by use of a markup language. Huge text datasets are marked up with standardized languages such as SGML, HTML and XML. Application specific, *ad hoc* markup languages are used in several cases, to group together logically coherent parts of the text. Emails, newsgroup postings and newspaper articles contain a certain layout due to their application specific markup.

As an example, consider product offers from Yahoo! Shopping[1] where an *ad hoc* markup language is used to point out the attributes of a product. An example document for a Yahoo! product, with all its details, is presented in figure 1. In this, the layout segregates the *title*, *price*, *description*, *mid* (merchant id), *merchantCategory* and a *ManualCategory i.e.* the manually assigned category picked from Yahoo!'s taxonomy. Figure 2 lists some product categories to which these semi-structured product documents need to be assigned. The title carries the key information about the product while the description helps a buyer know details about it. Fields such as *merchant category* display correlations with product categories; products tagged with certain merchant categories tend to be assigned only certain specific categories. Each merchant has a unique identifier called *mid*. On an average, a particular merchant can be expected to be dealing only with a subset of product types. This leads to a correlation between the *mid* of a product and its category. Also, a product with very high price is likely to fall in the category of *Jewelery*. A manual categorizer considers all these fields while choosing the appropriate category for a product. An automatic categorizer could benefit from simulating the behaviour of a manual categorizer. This simulation could be achieved by supervised learning on manually categorized products.

[1]http://shopping.yahoo.com

<**title**:Bernese Mountain Dog Notecards>
<**price**:9.95>
<**merchantCategory**:Dogs Bernese Mountain Dog Notecards>
<**ppath**:Home Garden Garage/Arts Antiques Collectibles/>
<**mid**:1013203>
<**description**:These full-color, high quality notecards, by portrait artist Ruth Maystead, capture the true essence of the animal. They come in a box of 6 cards with envelopes and measure 5 1/2"x4 1/4".>

Figure 1: Example product

Flowers Gifts Registry/Flowers
Flowers Gifts Registry
Flowers Gifts Registry/Party Supplies
Home Garden Garage/Arts Antiques Collectibles
Home Garden Garage/PETS
Electronics
Apparel/Shoes
Jewelry Watches

Figure 2: List of example categories

1.1 Our Goal

We outline two goals based on our observations from several example products, one of which is exemplified in figure 1.

1. The document is semi-structured and each field in the document carries information with specific relevance to its category. Values in the "Merchant Category" field exhibit preferential association with certain product categories. Products belonging to the "Jewelry Watches" or "Electronics" categories tend to have numerically higher values in the "price" field than those belonging to the "Flowers Gifts Registry" or "Home Garden Garage" categories. In its own way, each field exhibits similarity with the category. Moreover, fields can have different levels of importance in characterizing the category. Any categorizer built by pooling together all the tokens across all the fields in every product document, obviously misses out the information embedded in the fact that the document has a segregation between certain groups of tokens, in the form of fields. We seek a categorizer that exploits the explicit information present in the documents' layout structure. Rather than exploiting this structural information by fiat, our goal is to learn to do so on the basis of training data.

2. The *title* field of the product has some words which are key to determining its correct category. For example, in the product shown in figure 1, the word "Notecard" is more important for determining the category than say, "Dog". Further, fields such as *title* and *description* of the product have natural language structure. A plain word-count based feature space cannot capture the inherent structure due to natural language. Our goal is to exploit the the implicit information carried in the natural language structure of phrase structured titles.

1.2 Our Contribution

This paper has two main contributions, *viz.*

- We present a generic, structure-sensitive algorithm for categorization of documents whose fields are demaracated. The algorithm provides for one or more features to be associated with each field. It also has a learning mechanism by virtue of which, importance, in the form of weights, for each of the field features for categorization, can be learnt from a set of categorized documents.

- We present an algorithm that exploits the syntactic parse structure of phrasal constructs. Our algorithm associates weights with words in such constructs and exploits their natural language structure to boost weights of important words called *focus words*. In the particular case of product documents, we identify two instances of *focus words* in product titles *viz. brand names* and *product names*, that tend to have high correlation with the respective product categories. The weights associated with the words in a phrasal construct are learnt from example instances of phrasal constructs, marked with the corresponding *focus words*. The learning is accomplished by training a classifier that uses several linguistic features obtained from the text's parse structure.

Our categorization framework comprises the two algorithms mentioned above. We refer to this framework as SSCAT. Emails, newspaper articles, newsgroup postings, products' data and advertisements are example documents that fall under the class of documents addressed by SSCAT. Categorization of these documents into specified categories carries immense commercial and scientific value.

1.3 Related Work

Existing methods for automatic text categorization[1, 4, 5, 11] do not use the layout and natural language structure information of semi-structure documents for their categorization. In [8], the authors use the structural information such as layout and *look and feel* of web pages for clustering structurally similar pages. Toward better information extraction, [3] presents a method for automatic extraction of entities from a document based on its visual characteristics and relative positions of the entities in the document layout. The class of documents that we address have a layout structure that is more *ad hoc*, while the existing methods have mainly addressed documents written in standard mark-up languages. In [14], the authors have used a *bag-of-concepts* model for improving the performance of SVMs in text categorization. These concepts are obtained from the co-occurence of words with contexts and documents (as in LSI). There have also been attempts [2, 6] at using natural language processing for improving the information retrieval task. Unlike these approaches, we use natural language structure for identifying important words in sentences, leveraging the recent advances made in natural language parsing techniques [9, 10]. The lingustically motivated indexing methods [15], use phrases instead of words for IR. In our approach, however, we identify features for categorization based on the natural language phrase structure. In [13], the authors draw a conclusion that the use of linguistic features do no good for text categorization tasks. Unlike their results, we observe that the use of right kind of linguistic features obtained from the syntactic parse of the phrasal

constructs, significantly improves the categorization accuracies.

2. EXPLOITING DOCUMENT LAYOUT

We associate one or more features f_{i_x} with each field i in the document. Thus, $\forall i$ we have only f_{i_1}, which we represent simply as f_i. There are two arguments for each feature *viz.* the product $prod_j$ and a category cat_k. Thus, $f_i(prod_j, cat_k)$ is any real-valued characteristic of the tuple $< i^{th}$ field of $prod_j$, $cat_k >$. For instance, f_i can be any measure of similarity between the i^{th} field of $prod_j$ and the collection of all i^{th} fields of the products in cat_k from training dataset. Details of the specific features we used can be found in section 2.4. Our goal is to learn the relative importance of the different field features of the product for determining the correct product category. This amounts to learning a weight factor λ_i for each feature $f_i()$.

2.1 The Classification Model

We define two classes; C_+ is the class of (product, correct category) pairs and C_- is a class of (product, incorrect category) pairs. Given a (product, category) pair, we wish to assign it to one of the two classes. We use the Bayes rule and calculate the relevant posterior probability

$$\begin{aligned} P(C_+|prod_j, cat_k) &= \frac{P(prod_j, cat_k|C_+)P(C_+)}{P(prod_j, cat_k)} \quad (1) \\ &= \frac{1}{1+e^{-\xi}} \end{aligned}$$

where

$$\xi = log[\frac{P(prod_j, cat_k|C_+)}{P(prod_j, cat_k|C_-)}] + log[\frac{P(C_+)}{P(C_-)}] \quad (2)$$

The first expression on the right hand side of equation 2 is the likelihood ratio while the second expression is the prior ratio. Next we have to choose a particular form for the class conditional density $P(prod_j, cat_k|C_1)$.

2.2 Exponential model for class conditional distribution of product-category pairs

We model the joint distribution of $prod_j$ and cat_k given each class C_*, as an exponential distribution given by equation 3.

$$\begin{aligned} p(prod_j, cat_k|C_*) = \frac{1}{Z} p_0(prod_j, cat_k|C_*) \quad (3) \\ * exp(\sum_i \lambda_{*i} f_i(prod_j, cat_k)) \end{aligned}$$

where the λ_{*i}'s are the parameters of the model for class C_*, Z is a universal normalization constant, and the $f_i(prod_j, cat_k)$'s are arbitrary computable properties, or *features*, of the product-category pair $(prod_j, cat_k)$ - in this case the similarity measures over various fields. $p_0(prod_j, cat_k|C_*)$ is any arbitrary initial distribution, sometimes loosely referred to as the "prior". For instance, we may model $p_0(prod_j, cat_k|C_j)$ as a uniform distribution. It might also be derived (using the chain rule) from a conditional distribution. Substituting from equation 3 for $* = +, -$ into equation 1, we get the equation 4 for logistic regression.

$$\begin{aligned} p(C_+|(prod_j, cat_k) &= \frac{1}{1+e^{(-\underline{\lambda}\underline{f}^T+b)}} \quad (4) \\ &= \sigma(\underline{\lambda}\underline{f}^T + b) \end{aligned}$$

where $\underline{f}$ is a vector of the features and $\underline{\lambda}$ is a vector of differences between the class parameters corresponding to respective features (f_i's). b is a scalar and is given in equation 5.

$$b = log[\frac{P(C_+)}{P(C_-)}] + log[\frac{P_0((product, category)|C_+)}{P_0((product, category)|C_-)}] \quad (5)$$

To use the class conditional exponential model to estimate the probability that a given (product,category) pair belongs to class C_+, one need only calculate $p_0(product, category|C_+)$ and the values of the various features $f_i(product, category)$, and use equation 4. Thus using the model is straightforward. In particular, the most probable category $cat(prod_j)$ for a product $prod_j$ is determined by equation 6.

$$cat(prod_j) = argmax_{k\in[1,|CAT|]} P(C_+|prod_j, cat_k) \quad (6)$$

We have cast this n-class classification problem into a binary classification problem because logistic regression internally would solve the n-class classification problem using binary regressors.

2.3 Estimating parameters

The problem is of predicting a binary output $y \in C_+, C_-$ from inputs $(product, category)$. Consider N *random samples* $(y^t, prod^t, cat^t)$, $t = 1, 2....N$ (these random samples correspond to training instances), where $y^t = C_+$ iff cat^t is the correct category for $prod^t$. . Else $y^t = C_-$. The likelihood function is given as

$$\begin{aligned} L &= \sum_{t=1}^{N} logp(y^t|prod^t, cat^t) \\ &= \sum_{t=1}^{N} logp(y^t|\underline{f}^t) \\ &= \sum_{t=1}^{N} (y^t log\sigma(\underline{\lambda}\underline{f^t}^T) \\ &\quad + (1-y^t)log[1-\sigma(\underline{\lambda}\underline{f^t}^T)]) \quad (7) \end{aligned}$$

Equation 7 can be rewritten as equation 8.

$$L = \sum_{t=1}^{N} (y^t log\sigma(\underline{\lambda}\underline{f^t}^T) + (1-y^t)log\sigma(-\underline{\lambda}\underline{f^t}^T)) \quad (8)$$

This is a non-linear function of $\underline{\lambda}$ whose maximum cannot be computed in a closed form. Two simple iterative algorithms for computing these would be the gradient descent and the Newton's method.

2.4 Feature Engineering

We identify features for broadly two types of fields in making an appropriate choice for $f_i(prod_j, cat_k)$; (1) the *price* field which has numerical values and (2) text fields *viz.*, *title*, *description*, *mid* and *merchantCategory*, each of which comprises one or more tokens.

Consider a product $prod_j$ that needs to be categorized. For the *price* field, we get the value of the $f_{price}(prod_j, cat_k)$ feature using the following steps:
(1) For each category cat_k, we concatenate the *price* fields of

all the documents in the training set, that have been associated with category cat_k. We call this concatenated entity as $price_k$. (2) Using lucene[7], we build an index on the numeric values of $price_k$ for all categories k. (3) Based on the value $price(prod_j)$ in the price field of $prod_j$, we frame a lucene query QPRICE = $[price_j \times 0.9$ TO $price_j \times 1.1]$. Note that for fields such as price, a range of values is considered instead of individual values. Our observations mentioned in section 1 showed that range of prices are correlated with the product categories. (4) QPRICE, fired on the index of $price_k$ retrieves categories that have some documents in their training set with price values falling in that range. Further, these categories are ranked. (5) For each test product $prod_j$, we obtain $f_{price}(prod_j, cat_k)$ as the reciprocal of the rank of cat_k in the above retrieved set. If a particular cat_k did not figure in the retrieved set (which can happen if cat_k had no product in the training set with price in the range specified by QPRICE), $f_{price}(prod_j, cat_k)$ is set to 0.

For every other field i, we get the value of $f_i(prod_j, cat_k)$ in the following manner:
(1) We generate instances using the i^{th} field of each product in the training dataset and associate the category of that product with this instance. (2) Using these instances for training, we build a multinomial naive bayes classifier with Lidstone smoothing. (3) For each test product instance $prod_j$, $f_i(prod_j, cat_k)$ is obtained as the probability of cat_k given the i^{th} field of product $prod_j$. That is, $\forall prod_j$, $f_i(prod_j, cat_k) = Pr(cat_k | field_i(prod_j))$.

In section 3, we present a feature for the *title* field, which is sensitive to the natural language structure of the *title*. The method for obtaining that feature, selectively boosts the weights of certain tokens in the product field to obtain weighted word counts for the multinomial naive bayes classifier.

3. FEATURE ENGINEERING FOR PRODUCT TITLE

The example product shown in figure 1 is a "notecard" with an inscribed image of the bernese mountain dog. This is apparent from the product *title*. However, a naive bayes classifier that is built using a bag-of-words model gets misled by the presence of words such as "dog" and assigns the incorrect category, "Home Garden Garage/Pets", to the product. Categorization of a document solely driven by global statistics of its consituent words and oblivious of their local importance in the context of the product, is prone to such errors. However, the fact that "notecard" is more important in determining the product category than "dog" or "bernese" is revealed to a human reader by virtue of the natural language structure of the title. We emulate this behaviour by analyzing the syntactic parse of the title and associating levels of importance, in the form of weights, with the different words in the title.

3.1 Focus words of text units

We introduce the notion of *focus* of a well structured phrase or a simple sentence. Given a task involving a natural language construct, we define the *focus* of the construct as, words in the construct that are key to accomplishing that task. For the task of detecting the answer entity to a factoid question, the focus are those keywords that specialize to an answer in an answer passage. The question, *What is the capital of Japan?* has "capital" as its focus word. Given the task of categorizing products, the *focus* of the product title, "Bobby Pins with Pear Colored Stone-22 color choices" is the product type "Bobby pins". Thus, *product names* in the product titles are focus words. Consider another product title "3202-1950 color slim CD jewel case by Memorex". The product name in this case is "CD jewel case" and is a strong indicator of the actual category, "Electronics". However, the title carries another strong clue for categorization in the *brand name*, "Memorex". In fact, "Memorex" manufactures products that are exclusively associated with the Yahoo! category "Electronics". Thus, *brand names* in product titles can also be recognized as *focus words* for the categorization task. We thus identified two classes of focus words in the product title that have high correlation with the product categories. These are namely, the *product name* and the *brand name* of the product.

3.2 Approach to Focus Word Detection

We outline algorithms that exploit the syntactic structure of product titles for identifying the two types of focus words, *viz.*, brand names and product names. The algorithms are based on supervised learning – they learn from example instances of phrasal constructs (in this case titles), marked with the corresponding *focus words*. Dictionary lookup is our baseline algorithm for focus word detection; a dictionary lookup algorithm will, for instance, look up a dictionary of product names and mark as product names, all those words in the title that happen to be in the dictionary. However, we believe that the vocabulary of product names keeps expanding. Moreover, in a product title such as "Bernese Mountain Dog Notecard", a dictionary lookup algorithm will mark "Dog" as well as "Notecard" as product names, since both actually happen to be product names in different categories; "Notecard" is a product name in the "Home Garden Garage/Arts Antiques Collectibles" category while "Dog" is sold under the "Home Garden Garage/PETS" category. Therefore, tagging both these names as product names in the above title will be misleading again. Hence, there is a need to analyse the natural language structure of the product and determine the focus words based on learning from example instances. We also refrain from rule-based tagging of focus words, given the fact that the notion of focus word could differ across tasks and hence, rules will need to be constantly tweaked.

However, the most compelling reason for subscribing to a learning-based approach is the following. Classifiers such as the probabilistic classifiers, associate a score of support, such as probability, with an instance's belonging to a particular class. For this particular problem there are two classes - a class of *focus* words and a class of *non-focus* words. We have an instance for each word, with features derived from the position of the word in the syntactic parse tree of the title. For a word "w" in a product title, the classifier will output a probability $p(focus\_class|w)$. These probabilities could be used in one of two ways:

(1) "w" can be marked as a focus word iff $p(focus\_class|w) > \theta$, where θ is some threshold. Thus, the tagging of focus words becomes highly sensitive to the choice of θ. Two

words, one with probability slightly less than θ and another with probability slightly greater than θ are treated very differently. This can hurt, if there are more than one word in the product title that are *focus words*, albeit to different degrees and if the probability $p(focus\_class|w)$ for one of the words falls marginally below the threshold. For instance, the product title "Softalk Shoulder Rests Model no 808ST" has two words "shoulder" and "rests" as part of the product name, although "rests" is more important as a focus word. (2) $p(focus\_class|w)$ could be used as weight factors for the title words. Focus words will get a higher score of support. These weight factors could be used to get weighted word counts for the multinomial naive bayes classifier for the title field instead of the simple multinomial classifier for the title as was described in section 2.4. Thus, a *natural language sensitive* feature for the title field could be obtained and plugged into the main product classification algorithm that exploits the layout structure of the document.

We choose the latter of the two methods. We select logistic regressor as our classifier since it is a probabilistic classifier and known to be one of the best performing classifiers for a two class problem. Section 3.5 describes how we use the classifier in a modified feature for the title field that incorporates information from focus words.

We generated the training data for focus word identification by manually tagging around 10 thousand product titles from Yahoo!'s online shopping data. Within each title, words that stood for the product name and the brand name were marked out separately.

3.3 Learning to detect product words in titles

We describe how the syntactic parse of a phrase or sentence can be helpful in identifying the product name. We build a classifier, with features derived from the parse structure of the title. To parse the title phrases, we use the lexicalized version of the PCFG parser [10].

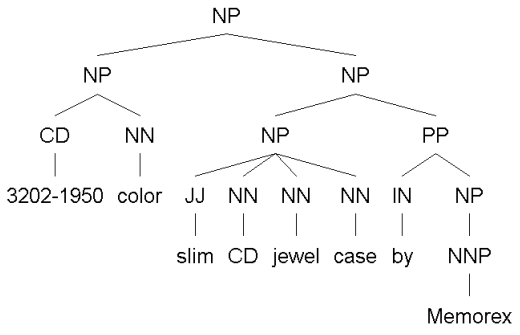

Figure 3: Example parse of a sentence as returned by the Lexicalized parser

The parse tree for the example phrase *3202-1950 color slim CD jewel case by Memorex* is shown in figure 3. Based on our observations on the parse structure of the title and the general trend in the position of product names in the parse structure, we tried constructing instances in two different ways.

1. In the first approach, only *heads* of phrases are considered as candidate product names. We generate an instance corresponding to every phrasal node in the parse tree returned by the lexicalized parser. For each instance, we identify features that capture the structure information. The features are: the height of the phrase, the part-of-speech of the head of the phrase, the phrase type of the preceeding phrase and the phrase type of the following phrase. *(NP (JJ slim) (NN CD) (NN jewel) (NN case))* is an example instance derived from the parse tree of figure 3. So is *(PP (IN by) (NN Memorex))* . The feature values for the example instance *(NP (JJ slim) (NN CD) (NN jewel) (NN case))* are listed in figure 4.

   The class labels for the training data were generated by assigning "product name" class to phrase instances whose heads were marked as part of the product name and "non product name" class to all other instances.

Feature	Description	Value
phraseHeight	height of the phrase from the root	*2*
phraseType	type of the phrase	*NP*
headPOS	part of speech of head	*NN*
nextPhrase	type of the phrase on the right	*PP*
prevPhrase	type of the phrase on the left	*NP*

Figure 4: Description of the features for the instance associated with the phrase *(NP (JJ slim) (NN CD) (NN jewel) (NN case))* **when an instance is associated with each phrase in the parse tree.**

2. In our second approach, we experimented with a different set of training instances; an instance was derived for every word/token in the text. We choose the following as features for each such instance: *whether the token is a head of any phrase (yes/no)*, *part-of-speech of the token*, *type of the tightest embedding phrase*, *the phrase preceeding the tightest embedding phrase* and *the phrase following the tightest embedding phrase*. Figure 5 shows the feature values for the token *case* in the example sentence of figure 3. The class labels for the training data were generated by assigning "product name" class to tokens instances which were marked as part of the product name and "non product name" class to all other instances.

3.4 Learning to detect brand names in titles

We use a learning based algorithm for identifying brand names in product titles in a manner very similar to that outlined in section 3.3 for product names. We derive an instance from every phrase in the syntactic parse of the title. The features for each instance were as follows: the height of the phrase, length of the phrase, the phrase type of the previous phrase, the phrase type of the next phrase, the words in the phrase and the part-of-speech tag of each of the words. We try two classifiers for this task - the logistic regressor and decision tree.

3.5 Modified Features for Title

Feature	Description	Example
isHead (yes/no)	is the token the head of the tightest embedding phrase?	*yes*
phraseType	type of the tightest embedding phrase	*NP*
posType	part of speech of the token	*NN*
nextPhrase	type of the phrase to the right of the tighest embedding phrase	*PP*
prevPhrase	type of the phrase to the left of the tighest embedding phrase	*NP*

Figure 5: Description of the features for the instance corresponding to *(NN case)* when an instance is associated with each token.

In section 2.4, we presented a uniform method using a multinomial naive bayes classifier for engineering features for fields such as title, description, mid and merchant category. The modified features for the title are brand name and product name that take into account the importance of focus words of the title phrase. We use the same multinomial naive bayes classifier as was suggested in section 2.4, except that we obtain weighted counts for words in the training dataset as well as in the test instance. The weights obtained are the $p(focus\_class|w)$ scores from the logistic regressor.

4. EXPERIMENTS

We present categorization experiments with our structure-sensitive framework on the products listed in Yahoo!'s online shopping data. The main experiments are related to

1. Learning and inferencing the weighted scores(also called boost-factors) associated with individual tokens for the titles composed in natural language. There are again two ways of assigning weights to tokens: a) based on how much the token is a product-name for the title and b) based on how much the token is a part of a brand-name.
2. Learning and inferencing with the log-linear model, whose features exploit the layout structure and using which we can predict the correct category label for a product.

Note that the output of the first part is used in determining the feature for the title field, which is plugged into the overall categorization algorithm (the second part). We report accuracies, precision and recall for each experiment, each averaged over 10 random runs. We also report variances along with the corresponding accuracies. Thus, an accuracy of $a \pm b$ means average accuracy for the experiment was a and the variance across the 10 accuracy results was b. In all the experiments, the ratio of training to test instances was kept at 60-40%.

4.1 Data Preparation

The data for our experiments came from more than 1 million Yahoo! products categorized into 67 categories. A sample from the list of categories was presented in figure 2. Each product comprised of exactly those fields as mentioned in figure 1. The training data for the focus word learning was created manually where a set of product titles were picked up at random. In these titles, the product name and the brand name were marked. For training the logistic regressor, the already existing manually categorized products were used.

4.2 Focus Word Detection in Titles

In this section, we present the results of experiments involving the product name learner and the brand name learner and compare our method against the dictionary lookup based baseline method(see section 3.2). The product and brand name dictionaries for the baseline algorithms were manually compiled from several thousands of Yahoo! products. Though we engineer the structure-sensitive features for title using logistic regressor, we also report accuracies with the J48 decision tree, to assess the suitability of the logistic regressor for this task. We used the Weka[16] implementation of LR[2] and J48. For product name detection, we compare the performance of LR and DTree against classification using Conditional Random Fields(CRFs).We used the implementation of CRFs given in [12].

4.2.1 Detecting Product Names (PNames)

We used the methods described in section 3.3 for learning to detect product names. We observe that J48 performs better than LR. When tokens are used as instances, CRFs do not perform as good as J48 and LR. Also, CRFs are not adept to learn with phrases as instances, since the sequence information is lost when phrases are used as instances. The results, reported over a 10-fold cross validation run, are shown in figure 6.

Categorizer	Accuracy(%)	PName(%) class precision	PName(%) class recall
DTree with phrases as instances	92.90±1.44	80.74±1.32	68.77±1.55
DTree with tokens as instances	90.98±1.14	72.89±1.21	56.70±1.10
LR with phrases as instances	91.72±1.76	78.09±1.88	61.83±1.60
LR with tokens as instances	90.81±1.25	72.00±1.39	56.41±1.19
CRFs with tokens as instances	88.66±0.97	70.73±0.88	32.77±1.05
Baseline (Dict. lookup)	60.15	48.23	69.43

Figure 6: Comparison of different classifiers and different feature sets on the task of Product Name detection.

In all these experiments, we observe that the 10-fold cross-validation accuracies are roughly equal to the training accuracies. This suggests that the classifiers are presumably not over-fitting the training data.

[2]We use the following abbreviations here. LR for Logistic Regressor, J48 for the decision tree, CRF for Conditional Random Fields, PName for the product name and BName for the Brand Name

All the learning based methods substantially outperform the baseline algorithm. The baseline algorithm shows somewhat high recall for the positive (i.e PName) class, owing to the fact the the product name dictionary is quite exhaustive. The pitfall in dictionary lookup for product names, which affects the overall baseline accuracies, was pointed out in section 3.2.

Figures 7 and 8 show the effect of the size of training data on the logistic regressor. While the precision saturates at lesser amount of training data, recall keeps increasing with increasing amount of training data.

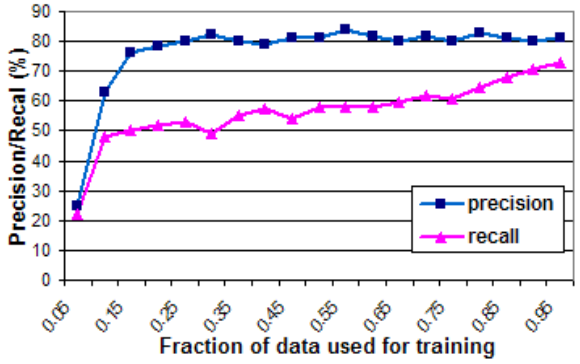

Figure 7: PName (positive) class statistics for LR as a function of the fraction of the data used for training.

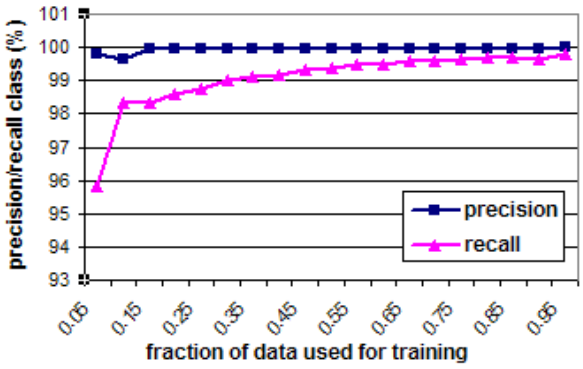

Figure 8: Non PName (negative) class statistics for LR as a function of the fraction of the data used for training.

4.2.2 *Detecting Brand Names (BNames)*

The results obtained for brand name detection, reported over a 10-fold cross validation run, are shown in figure 9. We used the training set with phrases as instances with the features as mentioned in section 3.4.

4.3 Categorization with SSCAT

When we incorporate the product-name and brand-name detection into the complete categorization framework, we give weights/boost-factors to individual tokens in a title rather than subsetting the tokens by applying a threshold to the classifier output. Both, the logistic regressor and the conditional random field model output the probability of class

Categorizer	Accuracy(%)	Precision(%)	Recall(%)
DTree with phrases as instances	99.69±1.61	97.64±1.48	99.60±1.77
LR with phrases as instances	98.38±1.17	94.32±1.26	99.10±1.10

Figure 9: Comparison of different classifiers on the task of Brand Name detection.

given the feature set for the instance. For these classifiers, the probability of the focus word class(say), given the instance is more interpretable as a weight for the token than the scores output by the decision tree. In the final setup, we experiment with the logistic regressor and the CRF classifiers for boosting tokens in the title. Since the tokens are only weighed relative to each other and no thresholding is done, the problem of low recall for focus word class as mentioned in section 4.2.1 is partly taken care of.

The SSCAT categorizer was tried with different lone features and the corresponding accuracies, micro-averaged precision and micro-averaged recall are reported in figure 10. The results show that the title feature based on boosted title, and in particular, boosted using LR model for product names with tokens as instances, helps categorization, more than the feature for the title based on simple word counts. Description also plays an important role in identifying the category to which the product belongs.

Feature for	Micro Averaged F1(%)
Title	74.5±1.23
Title boosted using LR for product names (token instances)	78.8±1.76
Title boosted using LR for product names (phrase instances)	77.8±2.25
Title boosted using CRF for product names (token instances)	74.4±2.12
Title boosted using LR for brand names (phrase instances)	76.3±0.56
Description	84.5±1.33
Merchant Cat	67.3±2.14
MID	77.3±0.67
Price	15±3.25

Figure 10: Comparison of performance with individual features

Categorizer	Accuracy(%)	Micro-avg(%) precision	Micro-avg(%) recall
Naive Bayes	85.0±1.6	87.6±1.7	76.3±1.2
SVM(one vs rest)	87.5±1.7	89.4±1.5	78.7±1.5
Our categorizer (SSCAT)	92.5±1.8	95.1±1.8	78.0±1.4

Figure 11: Comparison of SSCAT, with all field-features against other categorizers on same data (each averaged over 10 random runs).

Based on preceding observations, features on following fields were selected for the SSCAT classifier: *Title boosted using LR for product names, Title boosted using LR for brand names, Description, Merchant Category, MID and Price.* The resulting classifier gives accuracies that beat a bag-of-features based Naive Bayes and SVM classifiers. This result is tabulated in figure 11. All results are obtained over a

10-fold cross validation run. Figure 12 gives a plot of the accuracies obtained with SSCAT, as a function of the fraction of the data used for testing(remaining data is used for testing), each averaged over 10 random runs.

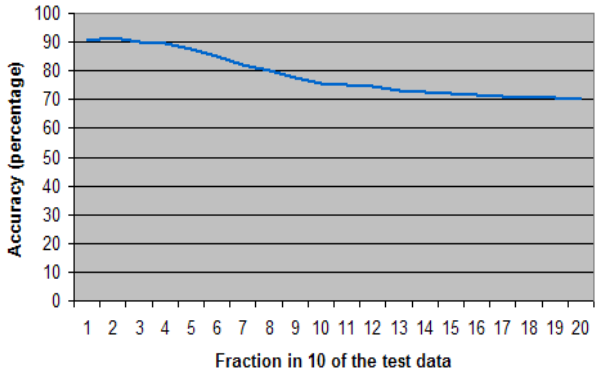

Figure 12: Plot of accuracies for the SSCAT against fraction of the data used for testing (remaining was used for training), each averaged over 10 random runs

5. CONCLUSIONS

We presented a categorization framework called SSCAT that takes advantage of the structure present in text to be categorized. We brought forth the existence of structure present in "semi-structured" text that can be exploited to improve the categorization accuracies. We presented two novel techniques, one for exploiting natural language structure and the other for the layout structure in text to be categorized. We used machine learning based algorithms in both the techniques. The categorization accuracy when categorizing Yahoo!'s shopping data with SSCAT is significantly better than the ones obtained with standard classifiers such as Naive Bayes and SVMs.

6. REFERENCES

[1] R. Basili, A. Moschitti, and M. T. Pazienza. A hybrid approach to optimize feature selection process in text classification. In *Proceedings of AI*IA-01, 7th Congress of the Italian Association for Artificial Intelligence*, pages 320–325, 2001.

[2] R. Basili, A. Moschitti, and M. T. Pazienza. NLP-driven IR: Evaluating performances over a text classification task. In *IJCAI-01, 17th International Joint Conference on Artificial Intelligence*, pages 1286–1291, 2001.

[3] R. F. Binyamin Rosenfeld and Y. Aumann. Structural extraction from visual layout of documents. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 203–210, 2002.

[4] C.-H. Cheng, J. Tang, A. Wai-Chee, and I. King. Hierarchical classification of documents with error control. In *Proceedings of PAKDD-01, 5th Pacific-Asia Conferenece on Knowledge Discovery and Data Mining*, pages 433–443, 2001.

[5] G. P. Cheong Fung, J. X. Yu, and H. Lu. Discriminative category matching: Efficient text classification for huge document collections. In *Proceedings of ICDM-02, 2nd IEEE International Conference on Data Mining*, pages 187–194, 2002.

[6] M. De Buenaga Rodríguez, J. M. Gómez-Hidalgo, and B. Díaz-Agudo. Using wordnet to complement training information in text categorization. In *RANLP-97, 2nd International Conference on Recent Advances in Natural Language Processing*, 1997.

[7] A. S. Group. Jakarta lucene text search engine. In *GPL Library*, 2002.

[8] S. Joshi, N. Agrawal, R. Krishnapuram, and S. Negi. A bag of paths model for measuring structural similarity in web documents. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 577–582, 2003.

[9] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the ACL Conference*, 2003.

[10] D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15*, pages 3–10, 2003.

[11] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. J. Watkins. Discrete kernels for text categorisation. In *Advances in Neural Information Processing Systems*, volume 13, pages 563–569. 2001.

[12] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[13] A. Moschitti and R. Basili. Complex linguistic features for text classification: A comprehensive study. In *ECIR*, pages 181–196, 2004.

[14] M. Sahlgren and R. Cster. Using bag-of-concepts to improve the performance of support vector machines intext categorization. In *COLING*, 2004.

[15] Strzalkowski, T. G. Stein, G. Bowden-Wise, and et al. Natural language information retrieval. In *TREC-7*, 1998.

[16] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, 2000.