Business Listing Classification Using Case Based Reasoning and Joint Probability

Sanjay Sood and Parijat Kat
AT&T Interactive
611 N Brand Blvd
Glendale, CA-91203
ssood@attinteractive.com, pkar@attinteractive.com

Abstract

One challenge of building and maintaining large-scale data management systems is managing data fusion from multiple data sources. Often times, different data sources may represent the same data element in a slightly different way.
These differences may represent an error in the data or a disagreement between sources on the correct value that best represents the data point. When the quantity of data managed and fused becomes sufficiently large, manual review becomes impossible, and automated systems must be built to manage data fusion. Some of the traditional solutions use simple voting theory, Dempster-Shafer theory, fuzzy matching and incremental learning. This paper presents a novel approach to data fusion in the domain of business listings. The task at hand, business listing categorization, suffers from conflicting and incomplete data from disparate data sources. Given the need for a high degree of accuracy in this task, we use a combination of case-based reasoning, joint probability, and domain-specific rules to improve data accuracy above other methods.

Introduction

Amount of electronic information in the modern world is being generated and stored at a high rate and this rate is rapidly increasing [1]. This electronic information constitutes the data sources which form the basis for businesses for domains such as social media, directory services, online entertainment, local business, etc. Individual entities enlist themselves under different service providers and service providers gather information about entities and try to enlist those in their system. There exist a large number of such data sources and a lot of their data overlap. There are many efforts by individual service providers to aggregate such information to provide a single entity view for the users. The challenge here is different sources can provide information about the same real-world entities; though, they may be represented differently and some may provide erroneous values. More importantly each service provider or source can try to classify an entity in a specific category based on the information which it has. While the source might publicly show some of the data attributes, the classification provided by it could be based on its own private data, private decision making algorithms and other factors provide erroneous values.

More importantly each source can try to classify an entity in a specific category based on the information which it has. While the source might publicly show some of the data attributes, the classification provided by it could be based on other private data and private decision making algorithms and other factors.

To resolve this ambiguity in data representation or classification, in practice, we can take each such classification as a vote and try to assign the classification with maximum votes as the primary classification in a master database. Another approach could be finding similarities in attributes in the master database and finding the associated classifications. However there are problems with these techniques. One good example is cable television guides, where the service provider tries to classify movies in specific genres. “Scary Movie” [2] though sounds like a horror movie because of the word “scary”, it is actually a comedy. If we try to classify this movie based on its name as an attribute, we will not find the right classification. If we have movie information provided by sources like IMDB or Rotten Tomato and use the classification votes from those sources, we might reach a better classification. However if we consider votes from such sources we need to rely on their accuracy.
Voting and confidence score based classification suffers from at least two problems. First, we do not know the algorithm or logic behind the classifications provided by each sources. Second, the source can have erroneous data.

Another approach of supervised learning [3] is difficult for such problem space because the data volume is too large and it is difficult to find a limited set of training data. The number of categories in a taxonomy tree could have more than 5000 categories and if the training set does not have enough sample data from each category, the classification does not find enough examples to reach a classification or leads to an erroneous classification.

However we have found that a similarity based classification using a technique like Naïve-Bayes[4] classifier and using a joint probability[5] based in combination to that improves the result significantly.

Problem Definition

Master object set M, be a set of real-world objects in the same domain such as business listing and directory entries. Each object is described by a set of attributes, and each object contains one or many values for each attribute. For example in the business listing domain, every listing has name, aka names, phone, address, reviews and ratings.

Some of these attributes could be empty or unavailable.

S is a set of data sources, each source having an element attribute format. Each source has a unique identifier as the source name and source code. For each object in M, a source can provide a set of records with a set of utes A, which may contain different values of a single attribute A_i, or different representations of the same value in A_i. Some of the values may not conform to certain data constraints and are subject to cleaning. Individual sources may have repetition of the same attributes. We clean such records and convert all the source records in the similar element-attribute relationship. Each data source also suggests a set of classifications for the objects in M. The classification could be of two types; "primary classification", PC and possible "other classification", OC. Hence, S is a composition of A, PC and OC.

Let's take an example we have a business listing $m \in M$ with a source $s \in S$ with unique source code "XYZ". Some examples of the attributes in S are business name, aka (also known as) name, phone number, address, reviews and ratings. We also get a set of classification c for m from s. However we do not know the rationale behind classifying m in category c, where $c \in PC \cup OC$. The source "XYZ" has its own algorithm and other data sources to determine such a classification. In the *Table 1* we show such a record.

Every source based on a general analysis has a confidence score. This score is a probability of the source being correct. Some of the sources are more reliable and some are less and some sources could be sacrosanct and could not be overridden because that has been manually reviewed.

Master Object id: 10000	
Source: XYZ Confidence Score: 0.6	
Attribute	Value
Name	Subway Cleaners
Aka	Subway Cleaners Inc
Phone	626-232-8059
Address	Chino Hills, CA
Primary Classification	Restaurant
Other Classifications	Cleaning Contractor

Master Object id: 20000	
Source: XYZ Confidence Score: 0.4	
Attribute	Value
Name	Sub Cleaners
Aka	Not Available
Phone	909-301-0121
Address	Chino, CA
Primary Classification	Dry cleaners
Other Classifications	Contractor

Table 1

Master Object id: 10000		
Primary Category		
Category Description	Source	Score
Cleaning Contractor	XYZ	90
Other possible Categories		
Category Description	Source	Score
Dry Cleaners	ABC	65

Table 2

Algorithm

In this paper we discuss a new algorithm: given a set S of independent data sources, and a taxonomy tree T, try to associate a set of primary category, $PC \in T$ and a set of permissible other categories $OC \in T$. Sample final results are in *Table 2*.

Taxonomy tree T is a tree structure of classifications for a given set of objects. At the top of this structure is a single classification, the root node that applies to all objects. Nodes below this root are more specific classifications that apply to subsets of the total set of classified objects. Each node in the taxonomy tree represents a category and that has a set of keywords K. For example the keywords associated with the node for category "restaurant" are "food, restaurant, cuisine, spice, taste". A sample taxonomy tree is shown in *Figure 1*.

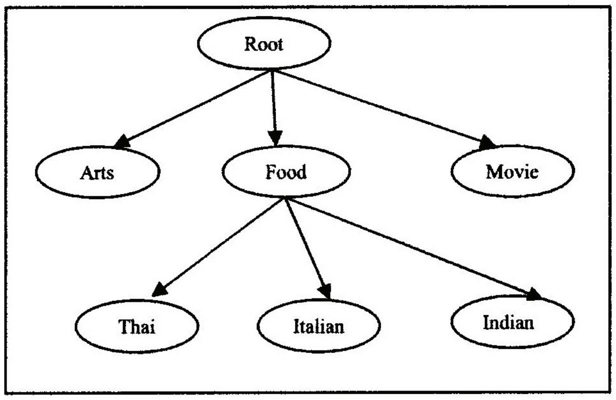

Figure 1

We compute a total classification score for the object by summing up similarity of attributes with the suggested category multiplied by the confidence score for that particular source. In other words, we compute the probability of a category suggested by a source to be true given the probability the source to be true and the probability of attributes provided by the source to be true. If we have source s_1 suggesting category C_1 for object o_1, we compute similarity between attributes of o_1 obtained from s_1 with the keywords k_1 for c_1 as $sim(a, c_1)$ where, $0 \leq sim(a, c_1) \leq 1$. If the confidence score for the source s_1, $CS(s_1)$ is the probability of the source s_1 to be true, by the theory of joint probability we can say that the category c_1 suggested by the source s_1 to be true is $sim(a, c_1) * CS(s_1)$. Hence by the theory of total probability we can say that the probability of the object o_1 to be in the category c_1 is

$$P\{(o_1 \in c_1)|s_1\} = \frac{sim(a, c_1) * CS(s_1)}{\sum_i sim(a, c_i) * CS(s_i)}$$

Similarity of attribute values to category keywords. In the domain of business listing most of the attributes are String variables and we try to find the similarity of an attribute with. Similarity in String attributes are determined using TF-IDF [6] algorithms.

In *Figure 2a* we describe the joint probability based algorithm. The simple voting algorithm is listed in *Figure 2b*.

```
getPrimaryCategory (Sources S, Master Object m,
            TaxonomyTree T) returns a category
  F is a list of categories f[0] to f[n];
  for (all attributes a[i] in A)
    for (all categories c[j] in T)
      f[i] = frequencies of different categories with
            similar attributes a
    end for
end for
avg = average(f);
F* is a list of categories f*[0] to f*[n];
for (all categories in F)
  if (f[i] > avg)
    add f[i] to F*;
end for

cat_clusters = createCategoryClusters (f*);

Table H is a map of categories and scores;

for (all s[i] in S)
  sim = 0;
  for (all a[i] in all attributes)
    K = getKeywords(s[i]);
    sim = sim + similarity (a[i], K);
  end for;
  confidence_score = getConfideneceScore(s[i]);
  if (s[i] in cat_cluster)
    confidence_score = confidence_score * GAMMA;
  score = sim * confidence_score;
  Put <category, score> in Table H;
end for;

sort table H on the highest score;

return the category with the highest score;

end function;

createCategoryClusters (List of Categories F) returns a
                         cluster of categories
  C is an empty list of category clusters;
  for (f[i] in F)
    boolean createNew = true;
    c = create a new cluster with the first element in F;
    for (f*[j] in F)
      if (isSimilar(f*[j], f[i]))
        add f*[j] to c;
        createNew = false;
        break;
      end if
  end for
  if (createNew)
    c= create a new cluster with first element as f[i];
    add c to C;
  end if
end for
return C;
end of function;

isSimilar(category1, category2) returns Boolean
  k1 = keywords from category1;
  k2 = keywords from category2;
```

```
    dist = taxonomical distance between category1
            and category2;
    if ((TFIDF(k1, k2) > BETA) and dist < D)
        return true;
    else
        return false;
end function;
```

Figure 2a

```
getPrimaryCategyByVoting(Sources S, Master Object m,
                        TaxonomyTree T) returns a category
    U = Unique sources(S);
    C = set of all the categories in U
    Table H is a map of categories and scores;
    for (c in C)
        Score[c]=sum of all the confidence scores from all the
                available sources for c;
        Put <category, score> in Table H;
    end for;
    sort table H on the highest score;
    return the category with the highest score;
end function;
```

Figure 2b

Experimental Results

We have run several set of experiments with the proposed algorithm comparing with other commonly available algorithms like voting and a simple Naïve-Bayes classifier. We also studied the changes in result with varying number of attribute and sources. The experiments are done with commonly available business listing entities collected from publicly available directory services like yelp.com, yp.com, allbusiness.com, superpages.com and each considered as a source system. On the first set of experiments we run 5000 business listings through a simple voting algorithm, Naïve-Bayes classifier and the proposed algorithm. The average number of source for each listing is 4.23. Also 127 business listings have no source suggested categories. Hence the voting algorithm cannot generate any primary or probable category as there is no available vote. The simple voting algorithm takes the category that occurs most number of times in all the sources for the given entity. We found that 4281 primary categories remain the same in all the three runs of the algorithms. The remaining 719 entities have different classifications generated by the new algorithm. By manual reviews we found that the new algorithm could not reach a conclusion based on insufficient or contradicting data in 123 entities. The 596 newly available categories are manually reviewed and found 440 of those to be more correct or better classification. We found 17 entities to have categories which are less acceptable than voting or Naïve-Bayes classification algorithms. This shows a significant improvement of result, considering 440 newly available categories are correct out of 596 changes in the categories, which a 73.8% accuracy on the changes and 8.8% on total number of listings. *Figure 1* shows a comparison of the results from three algorithms. The Amazon Mechanical Turk (MTurk)[7] is a crowd sourcing internet market place that enables computer programmers to coordinate the use of human intelligence to perform tasks which computers are unable to do. We also used MTurk to verify some of these results and the MTurk agreed to more than 90% of the new categories or has a overlapping result which has more than one categories, which has the category generated from the joint probability algorithm.

The next set of experiment shows the changes in result with availability of more attribute data are available. Primarily we compare results when more number of aka names and reviews for the listings are available. The results from increasing number of attribute values shows that the suggested algorithm works much with more number of attributes available (*Figure 3*).

The third set of experiments tries to find the scalability of the algorithm with more sources and more sources with incorrect classifications. We select a random category from the Naïve-Bayes classifier and create a new source, having the randomly selected category as the suggested category by that source. In *Figure 4* we show how the number of classification accuracy changes with more source systems and more noise. In this set of experiments the average numbers of noise or bad category sources were about 10%. Clearly we can see that both the voting algorithm and the joint probability based algorithm scales to classify more number of listings correctly when more number of source systems is available and only 10% of them are bad. However as the joint probability based algorithm uses more classifications from Naïve-Bayes classifier and computes similarities of attributes with those additionally available categories, the number of correct classifications are better than the simple voting algorithm.

To add some noise to the source systems, we also select some categories which are neither in Naïve-Bayes classifier, nor in the real sources. The objective is to find how the joint probability based algorithm can filter noises. In *Figure 5* we show how the algorithms scale when more noises are added to the system. We see that the voting algorithm fails sharply when noise is very high and greater than 40%. While the accuracy from joint probability based algorithm declines with more noise it could classify more than 5% listings with more accuracy. More importantly when the noise from source systems are less than 40%, which is a noise level close to real life data sources, the joint probability algorithm has done more than 12% better classification than the voting algorithm. More excitingly if the noise

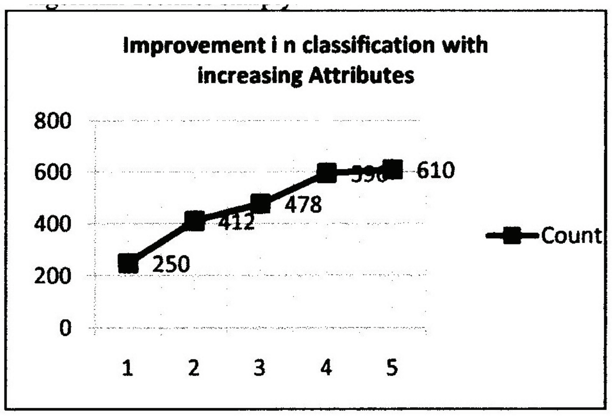

Figure 3

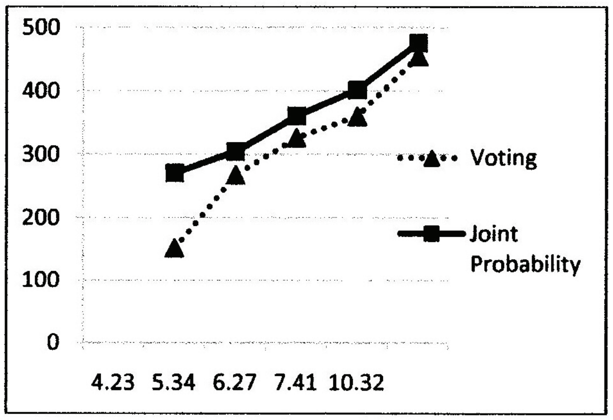

Figure 4

increases from 10% to 20%, the accuracy from joint probability algorithm does not decline much where the voting algorithm declines sharply.

Future Works

This work needs further exploration to see how increasing more the number of attributes in the source sets affects the result. Also we can explore how partially available data in the attributes could be used for similarity computation. We can also explore if some source system is an original source of the data or it is duplicating information from another source. If we have duplicate data from various sources, we will add additional voting factor to the computed category probabilities. There are several works to find record linkage in data fusion [8].

The future works need also need to include a good strategy for tie-breaking. If the scores are same for more than one category, we need to determine which will be a better candidate for primary category. In the current system we select the category which is a higher level on the taxonomy tree.

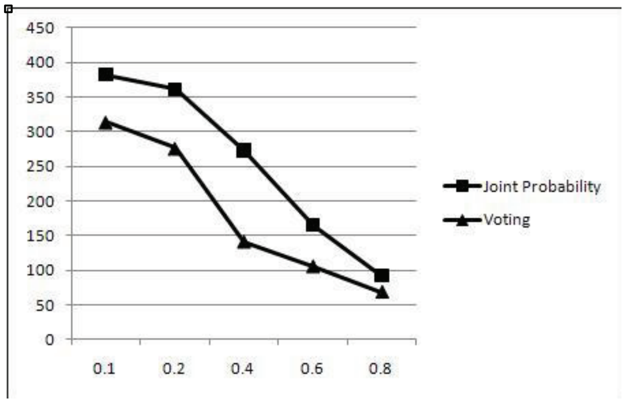

Figure 5

As we go higher level on the taxonomy tree, we get more generic categories. While this heuristics works in most of the cases, it fails for the cases which demands for more specific categories.

In a different ramification of the business listing efforts can potentially use internet crawlers to search for independent data for a given name, address, phone numbers for a less knows business. The searched information can include customer reviews and other critical information which could be extracted using several data mining techniques and can further improve the business listing classifications. For example, if the reviews which are found on internet search have too many keywords in the restaurant category like food, cuisine, taste, serving etc., we can try to associate the business to restaurant category.

Conclusion

This paper studies the problem of primary category detection in the presence different categories suggested by various sources and various attribute data presented by different sources. The key idea of our solution is to find the logic and justification of a classification provided by a source by finding similarities of such classifications with other available attributes from other sources and other suggested categories provided by other sources. We build clusters of categories from different sources and try to match the cluster with other attributes and finally pick the best category in top cluster. The experimental result studies show that this approach increases significant accuracy and the approach could be further improved by adding more attributes and other factors.

References

[1] J. Aslam, K. Pelekhov, and D. Rus. 1999. *A practical clustering algorithm for static and dynamic information organization*. SODA.

[2] Hiemstra, D. 2000. *A probabilistic justification for using tf.idf term weighting in information retrieval*. International Journal on Digital Libraries

[3] Hui Han , Lee Giles , Hongyuan Zha , Cheng Li , Kostas Tsioutsiouliklis. 2004. *Two supervised learning approaches for name disambiguation in author citations*. In IEEE joint conference on Digital libraries

[4] A. McCallum and K. Nigam. 1998. *A comparison of event models for naive bayes text classification*. In AAAI Workshop on Learning for Text Categorization.

[5] Marek J. Druzdzel. 1994. *Some Properties of Joint Probability Distributions*. In Proceedings of the 10th Conference on Uncertainty in Artificial Intelligence.

[6] Ho Chung Wu. *Interpreting TF-IDF term weights as making relevance decisions*. Journal ACM Transactions on Information Systems (TOIS). Volume 26 Issue 3, June 2008

[7] *https://www.mturk.com/mturk/welcome*

[8] Songtao Guo, Xin Luna Dong. 2010. *Record Linkage with Uniqueness Constraints and Erroneous Values*. VLDB 2010.