Chimera: Large-Scale Classification using Machine Learning, Rules, and Crowdsourcing

Chong Sun[1], Narasimhan Rampalli[1], Frank Yang[1], AnHai Doan[1,2]
[1]@WalmartLabs, [2]University of Wisconsin-Madison

ABSTRACT

Large-scale classification is an increasingly critical Big Data problem. So far, however, very little has been published on how this is done in practice. In this paper we describe Chimera, our solution to classify tens of millions of products into 5000+ product types at WalmartLabs. We show that at this scale, many conventional assumptions regarding learning and crowdsourcing break down, and that existing solutions cease to work. We describe how Chimera employs a combination of learning, rules (created by in-house analysts), and crowdsourcing to achieve accurate, continuously improving, and cost-effective classification. We discuss a set of lessons learned for other similar Big Data systems. In particular, we argue that at large scales crowdsourcing is critical, but must be used in combination with learning, rules, and in-house analysts. We also argue that using rules (in conjunction with learning) is a must, and that more research attention should be paid to helping analysts create and manage (tens of thousands of) rules more effectively.

1. INTRODUCTION

Classification is a fundamental problem in machine learning, data mining, and data management [20, 21]. Large-scale classification, where we need to classify hundreds of thousands or millions of items into thousands of classes, is becoming increasingly common in this age of Big Data. Such needs arise in industry, e-science, government, and many other areas.

So far, however, very little has been published on how large-scale classification has been carried out in practice, even though there are many interesting questions about such cases. For example, at this scale, how does the nature of the learning problem change? Would existing methods that employ manual classification, supervised learning, or hand-crafted rules work? In recent years crowdsourcing has emerged as a popular problem solving method [8], and has been applied to classification (e.g., [6, 29, 3, 11, 22, 13]). Can crowdsourcing work at such large scales? It is often informally understood that industrial practitioners employ hand-crafted rules to "patch" system performance. What kinds of rules do they write? How many, and who write these rules? Finally, how do we evaluate the output of classification at this scale? Answers to these questions can significantly advance the development of effective solutions to large-scale classification, an important Big Data problem.

In this paper we explore the above questions. We describe the product classification problem at WalmartLabs, a research and development lab of Walmart, the second largest e-commerce company in the world. This problem requires classifying tens of millions of product descriptions into 5000+ product types (e.g., "area rugs", "rings", "laptop bags & cases") over an extended period of time. Interestingly, at this scale many conventional assumptions seem to break down and existing methods cease to work.

For example, many learning solutions assume that we can take a random sample from the universe of items, manually label the sample to create training data, then train a classifier. At this scale, however, we do not even know the universe of items, as product descriptions keep "trickling in", a few tens of thousands or hundreds of thousands at a time, over months (while we need to get the product classifier up and working as soon as possible). We never know when we have seen "enough products", so that we can take a representative sample. This of course assumes that the overall distribution of items is static. Unfortunately at this scale it often is not: the overall distribution is changing, and concept drift becomes common (e.g., the notion "computer cables" keeps drifting because new types of computer cables keep appearing).

Further, even if we have the entire universe of items at hand, it turns out that taking a representative sample for certain product types is still extremely difficult. Consider the product type "ornament", which turns out to have numerous subtypes. A domain analyst would need to spend hours searching the Web to understand the different subtypes, in order to create a representative sample for ornament (that contains examples of all subtypes). The same goes for "handbags" and "computer cables". Such problematic types are quite common at the scale of 5000+ product types, thus making it extremely time consuming to create representative training data.

One would imagine that crowdsourcing can help slash this time, as it has been commonly used to create training data for learning [6, 29, 3, 11, 22, 13]. Unfortunately at this scale the conventional wisdom does not seem to work: to label a product, a crowd worker must navigate a very

Proceedings of the VLDB Endowment, Vol. 7, No. 13

large set (5000+) of product types and often has to choose among multiple types that sound very similar (e.g., "utility knives", "pocket knives", "tactical knives", "multitools" for item "Gerber Folding Knife 0 KN-Knives"). This is not the kind of micro-questions that most crowd workers prefer and are well suited for, namely questions that requires little training and can be answered with yes/no within minutes.

For the above reasons and more, we show that existing solutions *by themselves* do not work well for our problem: manual classification takes too long; learning has difficulties obtaining sufficient and representative training data, handling "corner cases", and dealing with distribution changes and concept drifts; rule-based methods do not scale to 5000+ types; and crowdsourcing cannot help label training data (see Section 3).

We then describe our solution Chimera[1], which combines learning, hand-crafted rules, and crowdsourcing to perform large-scale classification. In particular, Chimera employs multiple classifiers, both learning- and rule-based ones. It employs crowdsourcing to evaluate the classification result, flag problematic cases, then forward those cases to the in-house analysts. The analysts analyze the cases, write rules to address them, correct the labels, and in general incorporate feedback into the system, to enable continuous improvement over time. We show that Chimera is novel in four important aspects (see Section 4):

- Chimera uses both learning and hand-crafted rules (written by domain analysts) extensively.
- It uses a combination of crowdsourcing and in-house analysts to evaluate and analyze the system, to achieve an accurate, continuously improving, and cost-effective solution for classification.
- Chimera is scalable in terms of human resources, by using in-house analysts and tapping into crowdsourcing, the most "elastic" and "scalable" workforce available for general use today, and
- Chimera uses a human-machine hybrid algorithm that treats learning, rules, crowd workers, in-house analysts, and developers as "first-class citizen".

Overall, we make the following contributions:

- We show why many conventional assumptions and existing methods break down for large-scale classification.
- We describe Chimera, a solution developed and has been in use for years at WalmartLabs, which combines learning, rules, crowdsourcing, in-house analysts, and developers to perform accurate, continuously improving, and cost-effective classification.
- We describe a set of lessons that can be valuable for other similar Big Data systems. In particular, we argue that at large scales crowdsourcing is critical, but it must be used in combination with learning, rules, and in-house analysts. We also argue that using rules (in conjunction with learning) is a must, and that more research attention should be paid to helping analysts create and manage (tens of thousands of) rules more effectively.

[1]Chimera is a creature in Greek mythology that composed of the parts of three animals: a lion, a snake and a goat.

2. PROBLEM DEFINITION

We now describe the problem considered in this paper: classifying millions of product items into thousands of product types in a product taxonomy.

Product Items: A product item is a record of attribute-value pairs that describe a product. Figure 1 shows three product items in JSON format. Attributes "Item ID" and "Title" are required. Most product items also have a "Description" attribute. Both "Title" and "Description" are textual (see Figure 1). Some product items have more attributes (e.g., "Manufacturer", "Color"). Many, however, come with just the "Item ID" and "Title" attributes (this is the worst-case scenario for our classification system).

Millions of product items are being sent in continuously (because new products appear all the time) from thousands of Walmart vendors. In theory, we can ask the vendors to fill in a very elaborate set of attributes (e.g., "Title", "Color", "Manufacturer", "Weights", "Description", etc.) when describing a product. In practice this raises many problems: vendors may be reluctant to spend a lot of effort filling in the attributes; they may agree to fill in the attributes, but doing so with low accuracy; and this may take a long time. Consequently, we ask them to fill in at least the "Title" attribute, and optionally other attributes, as many as they are comfortable. This explains why product items typically do not have a very rich set of attribute-value pairs, as shown in Figure 1.

Product Taxonomy & Product Types: We maintain a very large product taxonomy, based on the Structured Commerce Classification initiative that Walmart and eBay spearheaded with GS1. The taxonomy is created offline using automatic, outsourcing, and crowdsourcing methods. It has more than 5000 mutually exclusive product types, such as "laptop computers", "area rugs", "laptop bags & cases", "dining chairs", "decorative pillows", and "rings".

The taxonomy is constantly being updated, with nodes being deleted, merged, modified, and new nodes being created all the time. Consequently, the set of product types is also constantly changing, and this significantly increases the complexity of our classification task. For the purpose of this paper, however, we will assume that the set of product types remains unchanged. Managing a changing taxonomy in a principled fashion is ongoing work.

Classifying Product Items into Product Types: Our goal is to classify each incoming product item into a product type. For example, the three products in Figure 1 are classified into the types "area rugs", "rings" and "laptop bags & cases" respectively. Our problem setting is distinguished by the following characteristics:

- We have a very large number of product types (5000+) to classify into, and we started out having very little training data (for learning-based classification). Creating training data for 5000+ product types is a daunting task, and for certain product types, it is not even clear how to create a representative sample of training data, as we will see later.
- We have limited human resources, typically 1 developer and 1-2 analysts. The analysts can be trained to understand the domain, but cannot write complex code (as they typically have no CS training). Such a

```
{
"Item ID": 30427934,
"Title": "Eastern Weavers Rugs EYEBALLWH-8x10 Shag Eyeball White 8x10 Rug",
"Description": "Primary Color: White- Secondary Color: White- Construction: Hand Woven- Material: Felted Wool- Pile Height: 1''- Style:
 Shag SKU: EASW1957"
}
{
"Item ID": 31962310,
"Title": "1-3/8 Carat T.G.W. Created White Sapphire and 1/4 Carat T.W. Diamond 10 Carat White Gold Engagement Ring",
"Description": "As a sincere declaration of your devotion, this gorgeous Created White Sapphire and Diamond Engagement Ring is fashioned
 in lustrous white gold. The engagement ring showcases a stunning created white sapphire at its center and a total of 24 round, pave-set
 diamonds along the top and sides of the rib-detailed band."
}
{
"Item ID": 17673919,
"Title":"Royce Leather Ladies Leather Laptop Briefcase",
"Description":"This handy ladies laptop brief has three zippered compartments. The first on the front is a deep pocket with a tucked-in
 hidden zipper. The second compartment has multiple interior pockets for a Blackberry/Palm Pilot, cell phone, business cards, credit cards,
 pen loops and much more. The final compartment is divided to hold your laptop computer and files all in one location. Topped off with two
 handles and a comfortable shoulder strap."
}
```


Figure 1: Examples of product items.

small team size is actually quite common even at large companies, because a large company often must spread its people over many projects. In any case, when classifying at such a large scale, doubling the team size is unlikely to help much. Instead, we believe the key is in using crowdsourcing, as we will discuss.

- Product items often come in bursts. There may be a sudden burst of hundreds of thousands of items that come in (either from a new vendor, or from an existing vendor who has just finished cleaning their items). We would need to classify these items fast, so that they show up on walmart.com as soon as possible. This makes it hard to provision for analysts and outsourcing people.

- We need very high precision, around 92% or higher (i.e., 92% of our classification results should be correct). This is because we use the product type of an item to determine the "shelf" on walmart.com on which to put the item, such that users can easily find this item by navigating the item pages on walmart.com. Clearly, incorrect classification will result in putting the item on the wrong "shelf", making the item difficult to find and producing a bad customer experience. We can tolerate lower recall, because items that we cannot yet classify will be sent to a team that attempts to manually classify as many items as possible, starting with those in product segments judged to be of high value for e-commerce. But of course we want to increase recall as much as possible.

We now discuss how the above characteristics make current approaches not well suited to solving our problem.

3. LIMITATIONS OF CURRENT APPROACHES

To solve the above problem, we can manually classify the items, use hand-crafted rules, or use machine learning. None of these solutions by itself is satisfactory, as we discuss next.

3.1 Manually Classifying the Product Items

We can manually classify the items using analysts, outsourcing, or crowdsourcing.

Using Analysts: We find that an analyst can accurately classify about 100 items per day (or 13 items per hour, assuming 8-hour workday). So classifying an item takes about 4-5 minutes. The reason it takes this long is because the analyst must understand what the item is, navigate through a large space of possible product types, examine them, then decide on the most appropriate one.

For example, given a product item, the analyst may first do research on what are the potential product types, normally by searching on the Web to get more detailed descriptions, pictures or any other related information. Some item titles can be very misleading. For example, without a lot of research, it would be difficult to determine that item "Misses' Jacket, Pants And Blouse - 14 - 16 - 18 - 20 - 22 Pattern" has the product type "sewing patterns".

After having a rough idea about what potential segments or product types an item belongs to, the analyst must navigate (using a visual tool) to a particular product segment, look at hundreds of product types in that segment, and decide which one seems the most appropriate. This is not easy, as there may be two or more product types that look very similar, or there is no suitable product type in our taxonomy yet. (Our taxonomy structure is not fully mature yet and taxonomists are actively updating or adding more product types.)

Take item "Dynomax Exhaust 17656 Thrush Welded Muffler" for example. The analysts could recognize it to be a muffler. However, the taxonomy may not yet have muffler as a product type. So the analysts need to look for the closest product type for muffler. This takes time as we have about 150 product types related to automotive, such as "automotive brakes" and "automotive bumpers". In certain cases we need to spend a lot of time finding the best product type. For example, we have multiple product type candidates "utility knives", "pocket knives", "tactical knives" and "multitools" for item "Gerber Folding Knife 0 KN-Knives". Which product type to select is difficult to decide. Different analysts may even have different opinions and they need to consult each other to make sure they are consistent in classifying the items.

Given the rate of 100 product items per day per analyst, it would take 200 days for a team of 5 analysts to manually classify 100,000 items. Thus, asking analysts to manually

classify incoming product items is clearly not a practical solution.

Using Outsourcing: Outsourcing means contracting with a team of people in another company (typically in a developing country) to do the work. This practice has been used extensively in industry for decades. In our problem context, however, outsourcing does not work well for two reasons. First, it is prohibitively expensive. Assuming an outsource worker charges a typical $10 per hour and also can classify 13 items, or roughly 77 cents per item. Classifying 100,000 items would incur $77K, and a million items incur $770K, an unacceptable cost to us.

Second, outsourcing is not "elastic", i.e., it is very difficult to quickly scale up and down the outsourcing team on demand. Recall that product items often come in bursts. If we keep a small outsourcing team, then when hundreds of thousands of items come in, we often cannot scale the team up fast enough to classify the items (contract negotiations take time, many outsourcing companies do not have tens to hundreds of people at standby ready to work at a moment's notice, and training new people takes time). On the other hand, if we keep a large outsourcing team at standby, then we have to pay a lot of money even when there is no or few product items to classify.

Using Crowdsourcing: Given the relatively high cost and the inelasticity of outsourcing, in recent years many companies have explored crowdsourcing for a wide range of data management tasks. In our case, however, a direct application of crowdsourcing does not work. As discussed earlier, classifying product items is a complex process that requires navigating a large taxonomy. This is not well suited to most crowdsourcing workers on popular platforms (such as Amazon's Mechanical Turk), who prefer micro tasks that require only a few tens of seconds to answer yes or no.

3.2 Learning-Based Solutions

We can create training data (i.e., ⟨product item, correct product type⟩ pairs), use it to train a learning-based classifier (e.g., SVM), then use the classifier to assign product types to items. In our context, however, this approach raises the following difficult challenges.

Difficult to Generate Training Data: As mentioned earlier, we started with very little training data. It turned out that generating sufficient training data for 5000+ product types is very difficult. To label just 200 product items per product type, we would need to label 1M product items. Our earlier discussion of the manual methods clearly shows that we cannot do this using either analysts, outsourcing (which would have cost $770K), or crowdsourcing.

One would imagine that we can try to obtain training data from another product taxonomy already populated with product items. For example, if our product type X corresponds to a type Y in some other product taxonomy, then we can just move all items associated with Y into X as training data for X. However, in practice, X and Y rarely match perfectly. For example, a taxonomy may refer to "child clothes" as clothes for children under 16, whereas another taxonomy may refer to "teenage clothes" as clothes for children age 12-18. So we cannot just move the items associated with Y into X, and filtering the items of Y is also very difficult.

Another idea is to have analysts and outsourcing workers write rules to generate training data. This would still be difficult and does not help generate representative samples as we will see below.

Difficult to Generate Representative Samples: A common method to generate training data is to take a random sample from the universe of product items to be classified, then label the sample (i.e., assigning correct product types to the items in the sample).

Our context however raises three problems. First, the set of product items that we have at any time is typically highly skewed. So some of the product types may be severely underrepresented in a random sample. For example, when taking a random sample of all product items in the Walmart catalog, more than 40% of the items in the sample are from the segment "Home & Garden". We end up having hundreds of thousands of "area rugs" and "stools" items, while having very few items for many other product types.

Second, even when we intentionally try to obtain a representative sample for a product type, it can turn out to be very hard, because we (analysts or outsourced workers) have no idea how to obtain such a sample. Consider for instance "handbags". How do we obtain a representative sample for this product type? All the items named "satchel", "purse" and "tote" are handbags and it is hard to collect a complete list of these representative items. Another example is "computer cables", which could include a variety of cables, such as USB cables, networking cords, motherboard cables, mouse cables, monitor cables, and so on. To obtain a representative sample for such a product type, an analyst may have to search the Web extensively, to understand all the various subtypes that belong to the same product type, a highly difficult task.

Finally, a related problem is that we do not even know the universe of all product items a priori. Recall that product items arrive regularly, with new vendors and new product types appearing all the time. As a result, the universe of all product items keep changing, and what we see at any time (i.e., the product items that have arrived so far) makes up just a fraction of this changing universe. This makes it difficult to judge the representativeness of the sampled data.

Difficult to Handle "Corner Cases": We often have "corner cases" that come from special sources and need to be handled in special ways. Such cases are difficult to handle using machine learning. For example, Walmart may agree to carry a limited number of new products from a vendor, on a trial basis. Since these products are new, training data for them are not available in the system, and it is also difficult to generate sufficient training data for these cases. As a result, we cannot reliably classify them using learning algorithms. On the other hand, it is often possible to write rules to quickly address many of these cases.

Another problem is that it is often difficult to use machine learning alone to "go the last mile". That is, we can use learning to achieve a certain precision, say 90%. But then increasing the precision from 90% to 100% is often very difficult, because this remaining "10%" often consists of "corner cases", which by their very nature are hard to generalize and thus are not amenable to learning. To handle such cases, we often must write rules, as we will see in Section 4.5.

Concept Drift & Changing Distribution: A problem that we hinted at earlier is that we do not know a priori the

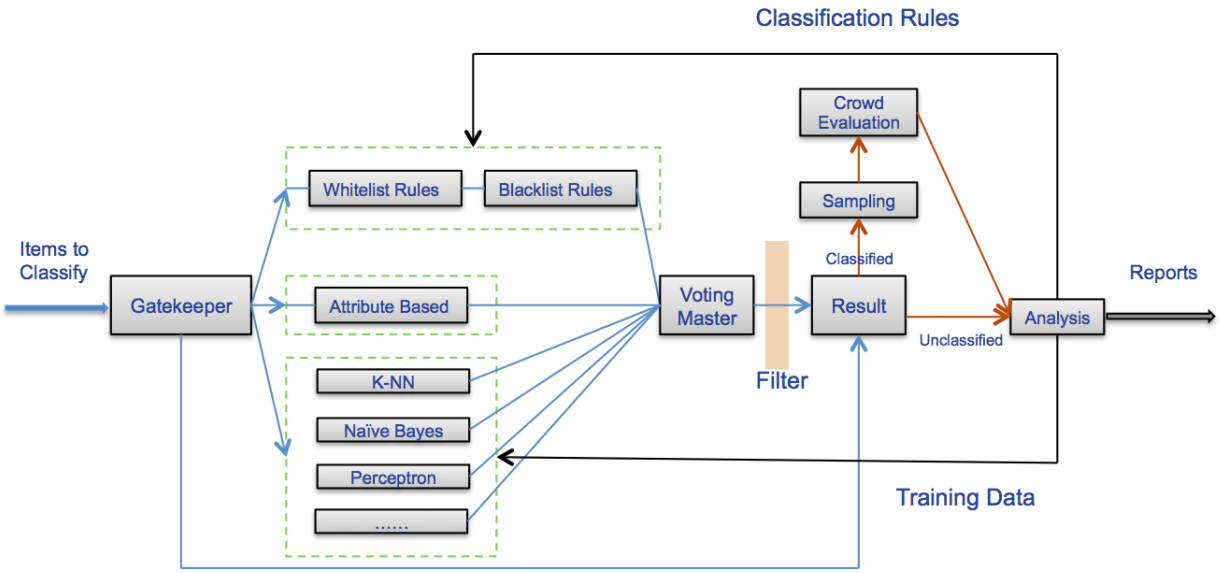

Figure 2: The Chimera system architecture.

items for each product type. In fact, the set of such items keeps changing, as new products appear. For example, the set of items for "smart phone" keeps changing, as the notion of what it means to be a smart phone changes, and new types of smart phone appear on the market. Thus, we have a "concept drift" (for smart phone). The distribution of all product items (over the product types) is also constantly changing. Today there may be a large portion of products in "Homes and Garden", but tomorrow this portion may shrink, in reaction to seasonal and market changes. The concept drift and changing distribution make it difficult to learn effectively.

3.3 Rule-Based Solutions

Instead of applying machine learning, we can just ask the analysts (and possibly also outsourcing people) to write rules to classify product items (see Section 4.5 for examples of such rules). But writing rules to cover all 5000+ product types is a very slow and daunting process. In fact, we did not find it to be scalable. In addition, we also waste labeled data that we already have for various product types (data that we can use to train machine learning algorithms).

4. THE CHIMERA SOLUTION

We now describe Chimera, our product classification solution that addresses the above limitations. Chimera uses a combination of machine learning, hand-crafted rules, developers, analysts, and crowd workers to form a solution that continuously improves over time, and that keeps precision high while trying to improve recall. (It is important to note that Chimera does not use outsourcing; we discuss the reasons for this in Section 5.)

4.1 Overall System Architecture

Briefly, Chimera works as follows:

1. Initialize the system using a basic set of training data and hand-crafted rules supplied by the analysts. Train the learning-based classifiers.
2. Loop:
   (a) Given a set of incoming product items, classify them, then use crowdsourcing to continuously evaluate the results and flag cases judged incorrect by the crowd.
   (b) The analysts examine the flagged cases, and fix them by writing new rules, relabeling certain items, and alerting the developers.
   (c) The newly created rules and the relabeled items are incorporated into the system, and the developers may fine tune the underlying automatic algorithm.
   (d) For those items that the system refuses to classify (due to a lack of confidence), the analysts examine them, then create hand-crafted rules as well as training data (i.e., labeled items) to target those cases. The newly created rules and training data are incorporated into the system, and it is run again over the product items.

As described, Step 2 not just classifies the incoming product items, but also uses crowdsourcing, analysts, and developers to provide a feedback loop into the system, to continuously improve its performance.

Figure 2 shows the overall Chimera architecture. Given items to classify, the Gate Keeper does preliminary processing, and under certain conditions can immediately classify an item (see the line from the Gate Keeper to the Result, and see the discussion in Section 4.3). Items surviving the Gate Keeper are fed into a set of Classifiers. We distinguish three types of classifiers: rule-, attribute-, and learning-based. There is one rule-based classifier that consists of a set of whitelist rules and backlist rules created by the analysts. There is one attribute- and value-based classifier that consists of rules involving attributes (e.g., if a product item has the attribute "ISBN" then its product type is "Books") or values (e.g., if the "Brand Name" attribute of a product item has value "Apple", then the type can only be "laptop",

"phone", etc.). The rest is a set of classifiers using learning: nearest neighbors, Naive Bayes, Perceptron, etc.

We use multiple classifiers because we have found that none of the classifiers dominates the performance all the time. Instead, since each classifier exploits a particular kind of information, by combining multiple classifier we can generally improve the classification accuracy.

Given an item, all classifiers will make predictions (each prediction is a set of product types optionally with weights). The Voting Master and the Filter combine these predictions into a final prediction. The pair ⟨product item, final predicted product type⟩ is then added to the result set (the box labeled Result in the figure).

In the next step, we take a sample from the result set, and ask the crowd to evaluate the sample. Briefly, given a pair ⟨product item, final predicted product type⟩, we ask the crowd if the final predicted product type can indeed be a good product type for the given product item. Pairs that the crowd say "no" to are flagged as potentially incorrect, and are sent to the analysts (the box labeled Analysis in the figure). The analysts examine these pairs, create rules, and relabel certain pairs. The newly created rules are added to the rule-based and attribute-based classifiers, while the relabeled pairs are sent back to the learning-based classifiers as training data.

If the precision on the sample (as verified by the crowd) is already sufficiently high, the result set is judged sufficiently accurate and sent further down the WalmartLabs production pipeline. Otherwise, we incorporate the analysts' feedback into Chimera rerun the system on the input items, sample and ask the crowd to evaluate, and so on.

If the Voting Master refuses to make a prediction (due to low confidence), the incoming item remains unclassified and is sent to the analysts. The analysts examine such items, then create rules and training data, which again are incorporated into the system. Chimera is then rerun on the product items.

4.2 The Novelties of Our Solution

As described, compared to existing work our solution is novel in four important aspects.

Use Both Learning and Rules Extensively: We use both machine learning and hand-crafted rules extensively. Rules in our system are not "nice to have". They are absolutely essential to achieving the desired performance, and they give domain analysts a fast and effective way to provide feedback into the system. As far as we know, we are the first to describe an industrial-strength system where both learning and rules co-exist as first-class citizens.

Use Both Crowd and Analysts for Evaluation/Analysis: We show how the step of evaluating and analyzing the classification results (to measure the system accuracy and provide feedback) can be done effectively using crowdsourcing and in-house analysts. At the scale that we are operating (5000+ product types and millions of product items), we argue that using crowdsourcing and in-house analysts is absolutely essential to achieve an accurate, continuously improving, and cost-effective solution.

Scalable in Terms of Human Resources: Our solution is scalable in that (a) it taps into crowdsourcing, the most "elastic" and "scalable" workforce available for general use today, and (b) it uses analysts. As the demand surges, we can scale by making heavier use of crowdsourcing and hiring more analysts, which is often far easier than hiring CS developers.

Treat Humans and Machines as First-Class Citizens: We show that to build an industrial-strength classification system on a very large scale, it is essential to consider not just automatic algorithms and rules, but also various classes of human, such as crowd workers, in-house analysts, and developers. All of these in essence need to be considered as "first-class citizens", and the solution needs to carefully spell out who is doing what, and how to coordinate among them. This is a clear demonstration for the need of hybrid human-machine solutions, as well as a novel case study on how such a solution works in practice.

In the rest of this section, we describe Chimera in more details, focusing on the above four novel aspects.

4.3 Initializing and Gate Keeping

We initialize the system with a set of training items and basic rules constructed by analysts (as many as we can create within a time limit). In the next step, incoming items are processed one by one. An incoming item x (e.g., in Excel or JSON format) is converted into a record of attribute-value pairs, then fed into the Gate Keeper.

The Gate Keeper can immediately make a decision regarding item x if (a) the title of x matches the title of an item y in the training data, or (b) a rule in the Gate Keeper can classify x with very high confidence (e.g., if x has attribute "ISBN", then x is of type "Books"), or (c) the title of x is empty. When (a) or (b) happens, the Gate Keeper sends item x together with the appropriate product type to Result. When (c) happens, the Gate Keeper also sends x to Result, but declines to make a prediction for x.

In all other cases, the Gate Keeper sends item x to the classifiers. We now describe the learning-, rule-, then attribute-based classifiers, in that order.

4.4 Learning-Based Classifiers

To classify a product item, our current learning-based classifiers consider only the product title. This is because e-commerce vendors typically take great care in creating such titles, packing the most important information about the product into the title (to make it as informative and catchy as possible to customers). Hence, we have found that focusing on the titles gave us a good start.

We have experimented with using product descriptions in learning-based classifiers, but found that they often contain "noise" that affects the classification results. For example, laptop descriptions often mention memory, hard disks, video card, CPU and even laptop cases, which are all different product types, thereby "confusing" the classifiers. We are currently working on managing such "noise" to effectively exploit product descriptions in our next-generation classifiers. Note that our other classifiers, rule-based and attribute/value-based ones, can and do exploit product descriptions and other attributes (e.g., "ISBN", "Brand"), as described in Sections 4.5-4.6.

To build a learning-based classifier, we manually labeled product titles to create training data. We regard each title as a mini document, and process it by stemming, removing stop words, and lower casing the entire title. We have explored using Naive Bayes, KNN, Perceptron, SVM, and

logistic regression, and found the first three work best. In what follows we briefly describe these classifiers.

Naive Bayes Classifier: This popular classifier is conceptually simple, easy to implement, provides fast classification, and is good with text data. In our case we started out using k-grams in the title with $k = 1, 2, 3$ as possible features. Eventually we use only bigrams (i.e., two-word phrases) as features because this produces the best performance. Given a product title t, we can compute $P(c|t)$, the probability that t has the product type c, for all product types, then return the top few product types that maximize $P(c|t)$ as candidate product types for title t.

Roughly speaking, let $f_1, \ldots, f_n$ be the (bigram) features derived from title t, then the Naive Bayes classifier computes

$$p(c|t) = P(t|c)P(c)/P(t),$$

where $P(t|c)$ is estimated as $P(f_1|c)P(f_2|c)\cdots P(f_n|c)$, assuming independence among the features. The probabilities $P(f_i|c)$ and $P(c)$ can be estimated from the training data, while the probability $P(t)$ can be ignored (as they are the same for all c).

KNN Classifier: This classifier assigns a product type to an item based on the types of similar items. It requires essentially no training, and achieves surprisingly good results. In particular, it guarantees that no incoming product item that already appears in the training data would be classified incorrectly. Specifically, given a product item x, we find the top k items in the training data that are most similar to x, rank the product types of these top k items based on their frequency, then return the ranked list as the classifier's prediction for item x. The similarity between two items is the weighted Jaccard measure between their titles (treated as sets of words), where the weights are computed based on the TF and IDF of the words in the titles.

Perceptron Classifier: This classifier uses a single-layer perceptron model with multiple outputs, and uses stochastic gradient descent for fast model training. Here we also use words in the items' titles as features, but perform feature selection using information gain: selecting the top 100 features based on information gain for each product type. This classifier can be trained fast with simple updates.

Finally, we have also experimented with SVM and logistic regression, but found that out-of-the-box versions of these methods underperform Naive Bayes and Perceptron. At the moment Chimera does not employ these methods. In ongoing work we are exploring whether better feature selection and parameter tuning can improve their performance.

4.5 Rule-based Classifiers

Whitelist- and Blacklist Rules: As discussed in Section 4.1, Chimera's analysts write many classification rules. We keep the rule format fairly simple so that the analysts, who cannot program, can write rules accurately and fast. Specifically, we ask them to write *whitelist rules* and *blacklist rules*. A whitelist rule $r \rightarrow t$ assigns the product type t to any product whose title matches the regular expression r. The followings for example are whitelist rules that our analysts wrote for product types "rings":

- rings? → rings
- wedding bands? → rings
- diamond.*trio sets? → rings
- diamond.*bridal → rings
- diamond.*bands? → rings
- sterling silver.*bands? → rings

The first rule for example states that if a product title contains "ring" or "rings", then it is of product type "rings". Thus it would (correctly) classify all of the following products as of type "rings":

- Always & Forever Platinaire Diamond Accent Ring
- 1/4 Carat T.W. Diamond Semi-Eternity Ring in 10kt White Gold
- Miabella Round Diamond Accent Fashion Ring in 10kt White Gold.

To make it easier for analysts to write rules, we assume that regular expression matching is case insensitive and that each regular expression starts on a word boundary (so "rings?" for example does not match "earrings"). Similarly, a blacklist rule $r \rightarrow NOT\ t$ states that if a product title matches the regular expression r, then that product is not of the type t.

Guidance on Rule Writing: We have developed a detailed guidance for our analysts on rule writing. Briefly, this guidance considers the following four important cases:

No training data: An important advantage of using rules is that they provide a good way to incorporate analysts' domain knowledge. Take product type "television" for example. Most analysts could easily create simple rules using regular expressions such as "televisions?","tvs?", "hdtv", "lcd.*tv", "led.*tv", etc. These rules can cover most items of "television" type (provided that item titles have reasonable quality; as a counter example of a low-quality title, it is nearly impossible to classify the item "ss glx gio s5660"). Thus, at the start, when there was no training data, we asked the analysts to create many basic rules based on their knowledge of the product types, to jump start Chimera. Later we also asked the analysts to create basic rules (as many as they are able) for product types with no training data.

Training data is too little and can mislead the classifier: It is also critical to use rules when we have limited training data for some product types. Classifiers trained on such limited training data often make many mistakes. Consider for example product type "ornaments". The initial training data for this type contained tens of examples, all of which contain the word "Christmas". Consequently, many Christmas items (e.g., Christmas trees) are classified as ornaments. In such cases, ideally we should try to obtain more training data. But this may be difficult and require too much effort to achieve in the short term. We deal with this problem in two ways. First, if a product type has too few training examples (currently set to 25, which we empirically found to work well), then we ignore these examples, thereby "turning off" the classifiers for this product type. And second, we ask the analysts to write rules for such cases, as many as they can.

Training data is not representative: In such cases whitelist-

and blacklist rules can provide very fast and effective solutions. Consider for example the product type "rings". We may have a lot of training examples for this type, but most of them mention "ring" in the title, and very few of them mention "bridal". As a result, we may not be able to classify "1 Carat Diamond Marquise Bridal Set in 10Kt White Gold" as a ring. Noticing this, an analyst can quickly write a whitelist rule "diamond.*bridal → rings" to address this and similar cases. Similarly, classifiers often misclassify "Miabella 1/4 Carat T.W. Diamond Solitaire 10kt White Gold Stud Earrings" as "rings". By adding the blacklist rule "earrings → NOT rings", we can quickly address this problem.

Corner cases: Finally, rules provide an effective and quick way to handle corner cases, such as new products from a vendor (accepted to walmart.com on a trial basis), or cases that prevent learning-based classifiers from "going the last mile", i.e., increasing the precision from 90% say to 100%.

In addition, we ask analysts to consider writing whitelist rules that have high precision, even if the recall is low. To achieve high precision, analysts may write blacklist rules that "prune" the coverage of a whitelist rule. For example, the whitelist rule "rings? → rings" covers also cases such as "Bluecell 50pcs 25MM Split Key Chain Ring Connector Keychain with Nickel Plated", which is clearly not a ring. To address this, an analyst may write the simple blacklist rule "key chain? → NOT rings". For the product type "rings", we have 12 whitelist rules and 76 blacklist rules (created over a period of months). Finally, the same product item may match multiple whitelist rules. For example, "Lucky Line 71101 Key Ring" matches both "rings? → rings" and "key.*rings → key chains".

Applying the Rules: Given an item x to be classified, we first apply all whitelist rules. Let S be the set of product types predicted by the rules that match x. Next we apply all blacklist rules to "prune" the set S.

In the next step we rank the types in S in decreasing likelihood of being the correct type for x. To do so, we use two observations. First, if the set of words (in the title of x) matched by a rule R_1 subsumes the set of words matched by a rule R_2, then R_1 is more likely than R_2 to have the correct product type. For example, consider "Lucky Line 71101 Key Ring". A rule R_1 for "key chains" may match "Key" and "Ring", and a rule R_2 for "rings" may match just "Ring". In this case the correct type is indeed "key chains".

The second observation is that in a product title (e.g., "LaCuisine 18pc Microwave Cookware Set KTMW18"), phrases that appear early in the title (e.g., Lacuisine, 18pc) tend to describe the characteristics of the product, whereas phrases that appear later in the title (e.g., Cookware Set) tend to refer to the product itself. So if a rule matches phrases later in the title, it is more likely to refer to the correct product type.

Formally, let W_i^s be the position in the title where the very first phrase that matches rule R_i starts, and let W_i^e be the position in the title where the very last phrase that matches rule R_i ends. Let t_i be the product types predicted by R_i. We have

- If $W_1^s = W_2^s$ & $W_1^e = W_2^e$, then $\text{rank}(t_1) = \text{rank}(t_2)$,
- if $W_1^e \leq W_2^s$, then $\text{rank}(t_2) \geq \text{rank}(t_1)$,
- If $W_1^s \leq W_2^s$ & $W_2^s \leq W_1^e$ & $W_1^e \leq W_2^e$, then $\text{rank}(t_2) \geq \text{rank}(t_1)$,
- If $W_1^s \leq W_2^s$ & $W_1^e \geq W_2^e$, then $\text{rank}(t_1) \geq \text{rank}(t_2)$.

Finally, we return the ranked list of types in S as the result of applying the rule-based classifier to product item x.

4.6 Attribute- & Value-Based Classifiers

These classifiers make predictions based on the presence of certain attributes or attribute values. For example, if a product item has attribute "ISBN", then classify it as "books". Other examples include "Director", "Actors", "Ratings", and "Screen Format" for movies, and "Console" and "Platform" for video games. As yet another example, if the brand attribute of an item has value "Apple", then we know that its type comes from a limited set of types, e.g., laptop, phone, etc. We ask the analysts to write such attribute- and value-based classification rules (in particular, our analysts have compiled a list of 20,000+ brand names together with associated product types). Such rules prove especially useful for certain product types, e.g., books, movies, musics, that share the same title (e.g., "The Hunger Games").

4.7 Voting Master and Filter

Given a product item x, the learning-, rule-, and attribute and value-based classifiers output ranked lists of predicted product types. The voting master then aggregates these ranked lists to produce a combined ranked list. It does so using either majority voting or weighted voting. Majority voting ranks product types based on their frequency in the classifiers' output, whereas weighted voting takes into account the weights of the classifiers (assigned manually by the developer or learned via training data).

Once the voting master has produced a combined ranked list of product types, the filter applies a set of rules (created by the analysts) to exert a final control over the output types. Note that analysts can already control the rule-based classifiers. But without the filter, they do not have a way to control the output of the learning-based classifier as well as the voting master, and this can produce undesired cases. For example, learning-based classifiers may keep misclassifying "necklace pendant" as of type "necklace" (because we have many items of type "necklace" in the training data that do contain the word "necklace"). As a result, the voting master may produce "necklace" as the output type for "necklace pendant". The analysts can easily address this case by adding a rule such as "pendant → NOT necklace" to the filter. As another example, at some point Chimera kept classifying some non-food item into "pizza". As a quick fix, the analysts can simply add a rule stating that if a title does not contain the word "pizza", then the type cannot be "pizza". We omit further details on the voting master and the filter.

4.8 Crowdsourcing the Result Evaluation

Motivations: We need to evaluate the accuracy of Chimera extensively during development as well as deployment. In particular, during deployment we must evaluate Chimera *continuously* for several reasons: (a) to detect cases where the output is not sufficiently accurate, so that the analysts and developers can improve those, (b) to ensure that the results we send further down the WalmartLabs production

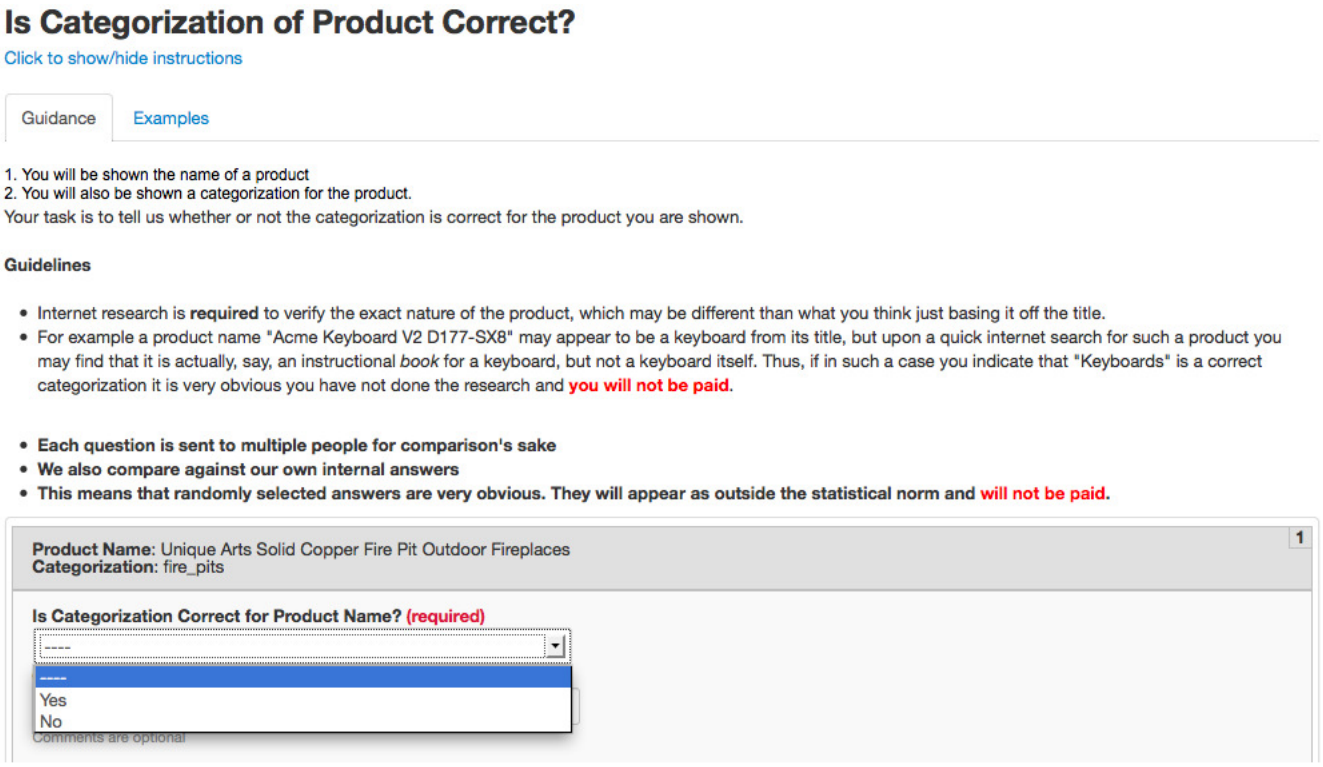

Figure 3: Sample question to crowd workers to verify the predicted type of a product item.

line are highly accurate, and (c) to detect accuracy deterioration: accurate cases that stop being so due to a change in the incoming data, the underlying algorithm, or the crowd.

Continuous evaluation is expensive. Our in-house analysts with good taxonomy knowledge can evaluate only 300 items per day on average. So if we have classified 1M items, and want to evaluate a 5% sample, that would be 50K items, which take 166 analyst workdays. Outsourcing is prohibitively expensive at the rate of $10/hour. Furthermore, since our items arrive in burst, it is very hard and expensive to provision and keep a large team of analysts and outsourcing workers on standby.

Crowdsourcing addresses the above problems and appears ideal for classification evaluation. Evaluation tasks can be easily distributed to multiple crowd workers working in parallel. These tasks are relatively simple and do not require much expert knowledge. In our experience crowdsourcing has proven scalable and even a large number of tasks can be finished in a short amount of time. We can crowdsource any time (and often do so during the nights). Finally, we do not have to maintain a large workforce dedicated to classification evaluation.

Sampling: To crowdsource the result evaluation, we perform two kinds of sampling. First, we sample over all the classified items, using a confidence level of 95% with a confidence interval of 2-3% to determine the number of items we should sample. Second, we do product type based sampling: from all product items classified into a type c, select a fixed number of items for evaluation. We use these two types of sampling to judge the overall performance of Chimera, and to pinpoint "places" that we need to improve.

Asking Crowd Workers: The samples contain pairs ⟨item, type⟩. For each pair we ask crowd workers to judge if the type can be a correct type for the item. Figure 3 shows such a sample question.

We use a third-party crowdsourcing platform that allows us to monitor worker performance and select only certain workers. We assign each question to two workers and get a third worker only if the two workers disagree (then do majority voting). Under this model, evaluating 1,000 items cost about $80.

Using Analysts to Analyze Crowd Results: All pairs ⟨item, type⟩ in the sample where the crowd says that "type" is not the correct type for "item" are flagged and sent to the in-house analysts. The analysts will verify whether this is indeed the case (thereby also verifying if the crowd is accurate), examine the problem, detect patterns, and write whitelist and blacklist rules to correct the problems. The analysts also manually correct the types of problematic items. Finally, the analysts report commonly observed patterns to the developers, who can do deeper debugging and adjusting of the underlying system.

It is important to note that to evaluate the crowd's performance, periodically the analysts will also evaluate a sample of the product items where the crowd agree with the predicted product types. Initially we also considered adding the crowd results directly to the training data of the learning-based classifier. This however posed a major issue. The periodic evaluation by the analysts suggests that the crowd accuracy is typically in the 97-98% range. So by adding their results to the training data, we add some incorrect training examples to the system. Later it may be very difficult to

"hunt down" these examples, should they cause problems. Worse, sometimes the accuracy of the crowd result drops to about 90% (e.g., when the incoming data is from a new source, or is unfamiliar in some way to the crowd). In such cases adding the crowd results to the training data can introduce a lot of incorrect examples. For these reasons, we now only use the crowd results to flag problematic cases that the in-house analysts will look at.

Handling Items that the Classifiers Decline to Classify: If the classifiers have low confidence, they will decline to classify the item. In such cases, we ask the analysts to look at the unclassified items, manually label a portion of them, and write rules. We then re-apply Chimera to these items, and repeat. Typically we can obtain fairly good accuracy (above 90%) with these items after a few days of such iterations.

5. EVALUATION & LESSONS LEARNED

The Chimera system has been developed and deployed for about 2 years. Initially, it used only learning-based classifiers. After an extended period of experimentation and usage, we found it difficult to achieve satisfactory performance using just learning alone (e.g., achieving only 50% precision at 50% recall). So we added rule-based and attribute- and value-based classifiers.

The system in its current form has been stable for about 6 months and have been used to classify millions of items (as of March 2014). Specifically, it has been applied to 2.5M items from market place vendors. Overall we managed to classify more than 90% of them with 92% precision. Chimera has also been applied to 14M items from walmart.com. Overall it classified 93% of them with 93% precision.

As of March 2014, Chimera has 852K items in the training data, for 3,663 product types, and 20,459 rules for 4,930 product types (15,058 whitelist rules and 5,401 blacklist rules; an analyst can create 30-50 relatively simple rules per day). Thus, for about 30% of product types there was insufficient training data (as of the above date), and these product types were handled primarily by the rule-based and attribute/value-based classifiers.

In terms of crowdsourcing, evaluating 1,000 items normally takes 1 hour with 15-25 workers. Evaluating 5,000 items and 10,000 items takes 3-6 and 6-12 hours, respectively.

In terms of staffing, we have 1 dedicated developer and 1 dedicated analyst. One more analyst can step in to help when there is a major burst of incoming products to be classified. This demonstrates that continuous ongoing accurate large-scale classification is possible with a "thin" staff, but only with the help of crowdsourcing.

The main lessons that we have learned with Chimera are the following:

- **Things break down at a large scale**: This has been demonstrated amply throughout this paper. For example, it may be very intuitive to think that crowd workers can help label examples to create training data. But it is not clear how they can do this given 5000+ labels. As another example, some of the labels (i.e., product types) have so many subtypes that it is very difficult to obtain a good representative set of examples for such labels.

- **Both learning and hand-crafted rules are critical:** Rules are not "nice to have". They play a critical role in Chimera (and in many other systems in industry that we know of) in injecting domain knowledge into the system, among others. It is highly desirable for the academic community to explore how to help analysts create rules accurately and fast, how to manage tens of thousands of rules (as in Chimera), and how to combine such rules with learning.

- **Crowdsourcing is critical, but must be closely monitored:** Crowdsourcing is not "nice to have". Without it we have no way to scale up performance evaluation and to handle unpredictable bursts of input in a time- and cost-effective fashion. But crowd performance can dip (to 90% or lower) on unfamiliar input data, and so needs close monitoring.

- **Crowdsourcing must be coupled with in-house analysts and developers:** For a system like Chimera, we need in-house analysts to examine problematic cases flagged by the crowd and to write rules, and we need developers to be able to debug the underlying system. While crowdsourcing has received much recent attention, the need for analysts has barely been studied.

- **Outsourcing does not work at a very large scale:** Chimera does not use outsourcing, as it is too expensive (and slow) at this scale. Eventually, we believe crowdsourcing may emerge as a new more effective "type of outsourcing".

- **Hybrid human-machine systems are here to stay:** While academia has only recently started exploring such systems, largely in the context of crowdsourcing, they have been used for years in industry, as evidenced by Chimera and other systems that we know. Such systems use not just crowd workers, but also analysts, and developers, and treat them as "first-class citizens". They have been quite successful, and deserve much more attention and closer studies.

6. RELATED WORK

Classification has been a fundamental problem in machine learning and data management [20, 21], and crowdsourcing has recently emerged as a major problem solving paradigm [8]. Many classification works have used crowdsourcing to obtain training data for learning [6, 29, 3, 11, 22, 13]. In Chimera, crowdsourcing cannot be used to obtain training data because crowd workers must navigate a very large label space (5000+ product types).

A few works have explored using crowdsourcing to complement the learning algorithm, for example when the learning algorithm is uncertain about predicting the label (e.g., reCAPTCHA project [1]). Some works have also used crowdsourcing for evaluation, but mostly for search relevance evaluation [2, 14] and entity matching [9]. In contrast, we use crowdsourcing to help evaluate the results of classification.

Machine learning techniques have been broadly used for document and text classification [15, 24]. As the most descriptive information of a product is textual (e.g., title or description), in our work we have applied a variety of learning based text classification techniques [18], such as KNN [30], Naive Bayes [16, 19], and Perceptron [23].

The Naive Bayes classifier [16, 19] has been proven very effective for text classification. Support vector machine (SVM) [10] is one of the most powerful classification techniques for document classification. However, a thorough evaluation of SVM in [17] shows that the performance of SVM classification on large-scale taxonomies is "still far from satisfactory".

Many techniques have been proposed to improve large-scale multi-label classification based on the hierarchical structure of the taxonomy or the structure of the categories [5, 27, 25, 26]. In our work however the taxonomy is relatively flat, with a large number of product types.

Besides learning-based models, rules are also used for classification. A rule-based classifier [7] uses a collection of "if ... then ..." rules to determine the item categories. Most of the previous research in rule-based classifiers focus on how to learn rules from the training data. In contrast, in Chimera we have analyst experts manually create classification rules using regular expressions.

Images could also be used for product classification when available [28]. However, automatically classifying a product item into thousands of categories based on images is very difficult. In [12] Kannan et al. proposed to use product images to help improve the text-based classification performance.

There has been relatively little published about large-scale classification systems in industry. The paper [4] describes LinkedIn's job title classification system, in which domain experts and crowdsourcing are also heavily used. Here crowdsourcing is used mainly for labeling "important" phrases for document classification (these labeled phrases are then exploited for learning). This process is similar to how we build classification rules. However, in our system we use crowdsourcing mainly for classification result evaluation. Furthermore, in-house analysts generate classification rules based on regular expression which is often more expressive than only phases.

The product categorization system at eBay is described in [25, 26], in which they have been focusing on classification based on the taxonomy structure and used a two-level classification approach. At the coarse level KNN is used and at the finer level SVM is used. Our classification strategy is different in that we combine multiple signals from different classifiers, including both learning- and rule-based ones. Furthermore, the system described in [25, 26] does not use hand-crafted rules and crowdsourcing like in Chimera.

7. CONCLUSION & FUTURE WORK

In this paper we have described the problem of classifying tens of millions of products into 5000+ product types at WalmartLabs. At this scale, we have demonstrated that certain conventional assumptions break down and existing solutions cease to work.

We have described Chimera, a solution developed at WalmartLabs that uses learning, hand-crafted rules, crowdsourcing, in-house analysts, and developers to successfully classify tens of millions of products with high precision and recall. We have also discussed a set of lessons learned. Our main messages are that all of the components – learning, rules, crowdsourcing, analysts, and developers – are critical for large-scale classification, and that it is important to explore further hybrid human-machine systems such as Chimera, which have proven successful in solving certain classes of real-world Big Data problems.

In ongoing work we are looking to develop tools that help analysts write rules better and faster, and to manage such rules. We are exploring methods to improve the current learning-based classifiers, as well as quickly generating good training data (e.g., using active learning). Finally, we are exploring how to use crowdsourcing for more types of tasks, such as writing simple classification rules.

8. REFERENCES

[1] *reCAPTCHA*. http://en.wikipedia.org/wiki/ReCAPTCHA.

[2] O. Alonso, D. E. Rose, and B. Stewart. Crowdsourcing for relevance evaluation. *SIGIR Forum*, 42(2):9–15, Nov. 2008.

[3] V. Ambati, S. Vogel, and J. G. Carbonell. Active learning and crowd-sourcing for machine translation. In *LREC*, 2010.

[4] R. Bekkerman and M. Gavish. High-precision phrase-based document classification on a modern scale. In *KDD*, pages 231–239, 2011.

[5] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, pages 163–171, 2010.

[6] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *ECCV*, 2010.

[7] W. W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123, 1995.

[8] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the World-Wide Web. *Commun. ACM*, 54(4):86–96, 2011.

[9] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *SIGMOD*, 2014.

[10] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *ECML*, pages 137–142, 1998.

[11] E. Kamar, S. Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *AAMAS*, 2012.

[12] A. Kannan, P. P. Talukdar, N. Rasiwasia, and Q. Ke. Improving product classification using images. In *ICDM*, pages 310–319, 2011.

[13] D. R. Karger, S. Oh, and D. Shah. Iterative learning for reliable crowdsourcing systems. In *NIPS*, 2011.

[14] J. Le, A. Edmonds, V. Hester, and L. Biewald. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation*, 2010.

[15] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR*, pages 37–50, 1992.

[16] D. D. Lewis. *Representation and Learning in Information Retrieval*. PhD thesis, 1992.

[17] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, 7(1):36–43, June 2005.

[18] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[19] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop*, 1998.

[20] T. M. Mitchell. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill, 1997.

[21] R. Ramakrishnan and J. Gehrke. *Database management systems (3. ed.)*. McGraw-Hill, 2003.

[22] V. C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, and L. Moy. Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322, Aug. 2010.

[23] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer-Verlag, 1996.

[24] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47, Mar. 2002.

[25] D. Shen, J.-D. Ruvini, and B. Sarwar. Large-scale item categorization for e-commerce. In *CIKM*, pages 595–604, 2012.

[26] D. Shen, J. D. Ruvini, M. Somaiya, and N. Sundaresan. Item categorization in the e-commerce domain. In *CIKM*, pages 1921–1924, 2011.

[27] C. N. Silla, Jr. and A. A. Freitas. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1-2):31–72, Jan. 2011.

[28] B. Tomasik, P. Thiha, and D. Turnbull. Tagging products using image classification. In *SIGIR*, pages 792–793, 2009.

[29] S. Vijayanarasimhan and K. Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR*, 2011.

[30] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR*, pages 42–49, 1999.