Building Re-usable Dictionary Repositories for Real-world Text Mining

Shantanu Godbole
IBM Research - India
shgodbol@in.ibm.com

Indrajit Bhattacharya*
Indian Institute of Science
indrajit@csa.iisc.ernet.in

Ajay Gupta
IBM Research - India
ajaygupta@in.ibm.com

Ashish Verma
IBM Research - India
vashish@in.ibm.com

ABSTRACT

Text mining, though still a nascent industry, has been growing quickly along with the awareness of the importance of unstructured data in business analytics, customer retention and extension, social media, and legal applications. There has been a recent increase in the number of commercial text mining product and service offerings, but successful or wide-spread deployments are rare, mainly due to a dependence on the expertise and skill of practitioners. Accordingly, there is a growing need for re-usable repositories for text mining. In this paper, we focus on dictionary-based text mining and its role in enabling practitioners in understanding and analyzing large text datasets. We motivate and define the problem of exploratory dictionary construction for capturing concepts of interest, and propose a framework for efficient construction, tuning, and re-use of these dictionaries across datasets. The construction framework offers a range of interaction modes to the user to quickly build concept dictionaries over large datasets. We also show how to adapt one or more dictionaries across domains and tasks, thereby enabling reuse of knowledge and effort in industrial practice. We present results and case studies on real-life CRM analytics datasets, where such repositories and tooling significantly cut down practitioner time and effort for dictionary-based text mining.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; J.7 [**Computers in Other Systems**]:

General Terms

Standardization

*Work done while at IBM Research - India

CIKM'10, October 26–30, 2010, Toronto, Ontario, Canada.

1. INTRODUCTION

Text Mining has been gaining ground as an important data mining application area in recent years. The text mining industry is still small [11] but is growing[1] quickly, along with the awareness of business insights hidden in unstructured data. The aim of text mining is to marshal huge amounts of unstructured data available in enterprises and externally, as on the internet, and use it to gain insights to solve real-world problems. Some typical industrial text mining applications [14, 1, 5] include 1) enhancing customer product and service experience by tracking what is being written in emails, surveys, and in social media, 2) enhancing churn and other predictive analytics models by leveraging features extracted from textual data for customer retention and extension, and 3) compliance monitoring applications in legal domains (e-discovery) and call centers (voice and rich media). A number of text mining product and service offerings from big and small companies exist that cite successful case studies. However, widespread adaptation and deployments are yet to happen.

In this paper we focus on dictionary-based text mining. Dictionary based systems are very common in information extraction[13], entity annotation[4], classification[5], and link analysis tasks. While statistical and model-based techniques thrive in academic research in text mining and NLP, real-world industrial products and services prefer tuned linguistic rule-sets and pre-packaged dictionary-based components, primarily because of the variance in the skill levels of end users. These components make text processing easily accessible, and facilitate browsing and understanding corpora. Practically, dictionaries are used to annotate (or 'tag') concepts, such as specific products or issues. Identified concepts can then be stored in entity databases for later use, or ingested by downstage analytical components. For example, correlation analysis can find insights such as problems strongly correlated with specific products, as in customer relationship management (CRM) applications.

On the flip side, constructing dictionaries has been called a practitioner's art [14], and requires experience and a lot of trial and error. Hand-tuning for measures such as precision and recall[1] is still the state of the art, as evidenced in the MUC conference series. The practitioner's domain knowledge becomes particularly important for tagging ab-

[1]Annual text mining industry conference `http://www.textanalyticsnews.com/text-mining-conference/`

stract concepts. Wordnet[2] is a popular resource that organizes general purpose concepts and entities for the English language. Similar Wordnets, customized and specific for various domains, would be invaluable for text mining. But it took years of manual effort to construct the English language WordNet. In this paper, we look at how tooling and automation can help text mining practitioners construct, use and evolve their 'private Wordnets', in the form of dictionary repositories.

While packaged dictionaries and taxonomies are available in many text mining products, the problem of building, adapting, and re-using domain-specific dictionaries remains a technical challenge requiring more interactive tooling than is currently available. When a perfectly constructed dictionary for a concept is readily available in a package, existing tools allow quick deployment. However, perfect dictionaries rarely exist for domain specific concepts. Accordingly, either existing dictionaries for related concepts need to be adapted, or completely new dictionaries need to be constructed for this new task. The first task that we focus on is dictionary construction. In typical text mining workflows, either domain experts know the set of dictionaries to create for the text mining task at hand, or they engage in a concept exploration and discovery phase[1]. Such concept discovery uses clustering (or some variant) to find coherent concepts in the text corpus and the experts then review the clusters and make dictionaries, rule-sets, classifiers starting from them.

The second task is dictionary adaptation across tasks, and possibly domains. While research in transfer learning [15] and domain adaptation [3] has been maturing in the past few years, direct application in real-world settings has only begun. Specifically, we are faced with the challenge of being able to leverage knowledge or supervision from earlier text mining tasks in new domains where exploration and discovery may be required. We want to develop a standard framework and methodology that practitioners can use in real commercial engagements for significant savings in time, effort, and cost.

The rest of our paper is structured as follows. We describe research challenges around dictionary building and re-use in Sec. 2, and review some related research work in Sec. 3. In Sec. 4, we present easy-to-use *dictionary construction and adaption tools* for practitioners providing a range of interaction and supervision modes. In Sec. 5, we show how to enable practitioners to *re-use earlier experience and supervision* in building dictionaries and quickly adapt them to newer tasks at hand. We apply the proposed techniques and show results on real datasets from voice of customer CRM analytics in Sec. 6. We recount our experiences as data mining researchers helping build commercial text mining service offerings in Sec. 7, and conclude in Sec. 8.

2. CHALLENGES

In this section, we set the context for our work around dictionary repositories. We discuss some novel research challenges arising in various real-world dictionary-based text mining tasks around interactively building, adapting, and reusing dictionaries. We then address some of these research challenges subsequently with our proposed framework.

[2]http://wordnet.princeton.edu/

2.1 Building dictionaries

Text mining practitioners have long recognized the benefits and tunability of rule-based data mining over complex statistical models [4] for information extraction, named-entity recognition, and text classification [7, 5]. High performance is practically achieved by carefully handcrafted rules and dictionaries perfected over time. In the light of these observations most text mining software products come with pre-packaged dictionaries, rule-sets, and taxonomies for typical application areas along with the ability to refine and enrich these pre-packaged components. We are familiar with the dictionary and rule-based extraction/annotation in commercial products from all leading text analytics vendors, as well as other freely available workbenches like GATE[3], Rapidminer[4]. However there are significant opportunities in improving their ease of use and range of supported operations for dictionary construction, modification, and re-use.

It has been recognized that verification is an easier text mining task than specification[1] for users. Currently, constructing dictionaries is clearly a tedious task of specification. How can we best *assist* the practitioner in constructing these dictionaries, in various scenarios where he may or may not be aware about the domain concepts ahead of time? This is an '*enabling the human in the loop*' challenge in the context of dictionary construction. What are the supervision (or semi supervision) models for these scenarios? For 'uninformed' users, can interactive modes of supervision work better than offline supervision? We pose the dictionary construction task as a ranking problem. To enable the user to provide feedback on a ranking with multiple seeds, how should the ranking be 'explained' to the user? Usually several concept dictionaries need to be constructed for a dataset. Are there benefits to constructing them simultaneously, rather than sequentially? We refer to all these questions collectively as the interaction challenge for dictionary construction.

2.2 Re-using dictionaries

An area of long standing concern for practitioners has been the ramp-up time for new text mining tasks and engagements. Reducing this time as much as possible is one of the driving factors behind pre-packaged components shipped with commercial products. Such components are often in the form of pre-built sets of dictionaries (among other components) standardized across tasks and industries like customer satisfaction (CSAT) analysis or opinion mining, and telecom, finance, or insurance respectively. These dictionaries, while useful, tend to prefer precision over recall and require significant manual customization before they can capture special characteristics of the new data at hand. Re-using dictionaries thus emerges as the second main challenge.

Allowing practitioners and domain experts to re-use their own or others' effort in past tasks and engagements can significantly speed up text mining. Recent advances in transfer learning[15] assume significance for this problem. Domain adaptation is another relatively recent related research area [3, 8] that considers the problem of reuse across domains. The recent framework of cross-guided clustering [2] aims at allowing re-use where supervision from earlier tasks is used to guide unsupervised concept discovery in newer tasks. Ad-

[3]http://gate.ac.uk/

[4]http://rapid-i.com/

ditional problems arise for dictionary adaptation when a single coherent concept in one dataset needs to be split up into multiple concepts in a new dataset. Vocabulary differences also lead to adaptation challenges. A dictionary may be relevant for a new domain even when none of the dictionary words exist in the new vocabulary, as for a Car Models dictionary when transferred from one manufacturer to another. How can a dictionary ontology be adapted for a new domain? What are the best interaction models for dictionary adaptation? These are some research challenges that arise in the dictionary adaptation and reuse problem, some of which we address in our framework.

3. RELATED WORK

There has been some work in the areas of dictionary building[9] in literature and in practice. A study of machine learning methods for information extraction tasks can be found in [4]. Bootstrapping has been applied to automatic dictionary building for information extraction [6, 13], where seed words are used to infer common extraction patterns. These patterns in turn help find words and phrases similar to the seeds and these two steps iterate and boost each other. There has also been work on unsupervised grouping together of word senses in a corpus [12]. These approaches are similar to the context based synonym finding we present in Sec. 4. Google Sets[5] is also related to dictionary building, but makes use of structured tables, possibly in addition to unstructured text. Our work differs in the interaction mechanisms and explicit feedback semantics.

Commercial products and free text mining frameworks enable dictionary-based text mining but simply allow manual dictionary construction. They rely on the human expert for the entire dictionary construction and adaption steps, and are not geared to handle noise and any corpus specific characteristics. In contrast, we present a complete framework for interactively building dictionaries from scratch and also adapting and re-using them.

The research area of transfer learning[15] is motivated by the need to transfer various types of supervision from one learning task to another. Improving accuracy of supervised classifiers by learning on data in a different domain [17, 16] is the most common form of transfer learning. Domain adaptation[3] is a specific setting where information extraction tasks like named entity recognition are adapted across domains using a *pivot* vocabulary that remains common across the domains. Another example is sentiment classifiers being adapted from the movies domain to the shopping domain[8] in online reviews. All these allude to scenarios where the earlier and the current tasks are both supervised in nature. Text mining engagements on the other hand are likely to be unsupervised or at best semi-supervised in nature, with concept discovery forming an important initial phase of the process where the practitioner determines what text mining components (dictionaries, rule bases, classifiers) are needed. Cross-guided clustering [2] (CGC) aims at allowing re-use where supervision from earlier tasks is used to guide unsupervised concept discovery in newer tasks. CGC is intended to be used when a set of supervised source partitioning (dictionaries) guides exploratory clustering in a target dataset. Manual intervention is needed for CGC in real-world settings – the human expert incorporates supervision by selecting dictionaries at the right level of *granularity* as source partitions. We will see examples of such supervision in Sec. 6. Mihalkova et al [10] have considered the problem of transfer across relational domains with minimal target data. This work is also relevant for multi dictionary adaptation across datasets.

We present real-life practical applications following the spirit of transfer learning for the dictionary-based text mining setting. Our work emerges at the junction of dictionary building and transfer learning in real-world commercial text mining settings, with a focus on practical interaction models.

[5] `http://labs.google.com/sets`

4. BUILDING DICTIONARIES

In this section, we focus on the task of building concept dictionaries for annotating a collection of documents from a particular domain. We first formally define the task of dictionary construction, and propose various approaches for it using different types of supervision from the user. Note that we do not look to automate the task of dictionary construction. We believe that manual intervention is critical for the task. Our goal in this paper is to provide tools that facilitate a user in the task of dictionary construction.

We define a dictionary $D = \text{Dict}(C, X)$ as a set of words that refer to or describe a semantic concept C in a document collection X. We will also use the term '*synonyms*' loosely to mean related words about C. Examples of concepts we would like to capture are mentions of problematic car-parts in agent notes in an automobile company's call center, or mentions of agent attributes like expertise or accent in customer surveys of a telecom company's helpdesk. Note that named entity lists like product parts, subscription plans, dealer names, etc. are naturally seen as concept dictionaries. The dictionary construction problem is then defined as:

Given a semantic concept C and a document collection X using a vocabulary V, return a ranking over words $w \in V$ such that words that refer to concept C appear higher in the ranking than words that do not refer to C.

A user inspects the ranking that is returned and decides which words actually refer to C and should be included in $\text{Dict}(C, X)$ and which words to discard. The general assumption is that 'critiquing is easier than constructing' — the user may not be able to create a dictionary from scratch, but when he recognizes a word from the dictionary when he sees it. Providing him with a good ranking according to C makes the dictionary construction task significantly easier for him since he needs to inspect a smaller set of words from the vocabulary. To aid the user in recognizing a dictionary word, some evidence explaining the inclusion of a word in the ranking is often useful. This is similar to providing snippets alongside with search results. We will return to this issue later in the section. In general, bi-grams and longer phrases may also be included in a dictionary, but in our experience 3-grams are usually sufficient for most scenarios. In the rest of this paper, we will use the term 'words' to refer to all these n-grams and phrases.

A critical issue in dictionary construction is specifying a semantic concept C. For general concepts in the English language, this may be done by referring to a node in the Wordnet hierarchy, and the intended dictionary would consist of all words in the synset corresponding to that node.

Unfortunately, the vocabulary of Wordnet is restricted to general English words, and it is not useful for specialized concepts and words that arise in domain specific uses. In section Sec. 5, we discuss how specialized semantic structures or repositories may be constructed gradually over time. In this section, we look at approaches to specify concepts in the absence of such a repository, or to specify new concepts not currently contained in an existing repository. For each of the different supervision models, we discuss algorithms for dictionary construction over document collections.

4.1 Single Seed Word

In the simplest model for concept specification, the user provides a single word w^s as an illustrative example of words corresponding to a concept C. For an automobile related dataset, the user may refer to the *Car Parts* concept using the word 'engine'. We will call w^s a seed word for C. The dictionary construction task then becomes to rank words in V according to 'semantic similarity' to the seed word w^s.

The two primary tasks that need to be addressed to solve this problem are defining the representation of a word $w \in V$ and a similarity measure $\text{sim}(w_1, w_2)$ between two words w_1 and w_2 based on their representation. Determining semantic similarity of words has a long history in natural language processing [9]. In the absence of background semantic knowledge, the basic assumption is that words that are used in similar local contexts over all documents in the collection are similar in meaning for that collection. Accordingly, we use a representation that captures the context of the words, and a similarity measure that compares contexts.

We capture local context around a word w occurring in a document by considering a *context window* of length l centered around w and considering words that appear within the context window. Each word w is then represented using a weight vector $WV(w)$ over words in vocabulary V, where the weight $WV(w, w')$ for any word w' captures the number of times w' has appeared in context windows around w over all documents in the collection. The counts are converted to normalized TF-IDF weight vectors[6] as usual for documents. To measure the similarity $\text{sim}(w_1, w_2)$ between two word representations, we use cosine similarity that takes the dot product of the two weight vectors $WV(w_1)$ and $WV(w_2)$. We will use this representation and similarity measure for words for all the supervision models and algorithms that follow. In fact, all the algorithms assume that the document collection in question has been pre-processed and the weight vectors for all words in the vocabulary are computed.

The algorithm for building a dictionary given a seed word w needs to rank the words w' from W in descending order of $\text{sim}(w, w')$. This can be performed efficiently using an inverted index as shown in Figure 1. In addition to the seed word w^s, the algorithm takes as input the maximum length k of the ranking to be returned and the minimum required similarity t with the w^s for a word to be included in the ranking. Also observe that the threshold t' determines a trade-off between accuracy and efficiency. As evidence to help the user decide which words from the ranking to include in his dictionary, it is possible to provide the highest weighted context words that lead to the similarity between w^s and a returned word.

[6] `http://en.wikipedia.org/wiki/Tf-idf`

```
    Algo BuildDict(Word w^s, Int k, Double t)

1.  Initialize candidate set CS to empty set
2.  For all words in WV(w^s)
3.      If WV(w^s, w') > some threshold t'
4.      Add w' to CS

6.  For each word w' in CS
7.      Compute similarity sim(w^s, w') with w^s
8.      Reject w' if similarity below threshold t
9.  Sort remaining words by similarity and return top k
```

Figure 1: Dictionary building algorithm with one Seed Word

4.2 Set of Seed Words

In general the user may provide not one but a set of keywords for specifying a concept C for constructing a dictionary. For example, for the *Car Parts* dictionary in our earlier example, the user may be able to provide an initial set containing the words 'engine', 'tires', 'brakes', 'gear' etc. We will call such a set a seed set S for C.

The approach in 4.1 can be generalized to accommodate a set of words as seed. First, however, the semantics of a seed set needs to be defined unambiguously. Returning to Wordnet for general purpose English words, each word w_i^s in the seed set S may be assumed to correspond to some node or synset C_i in the hierarchy. Ideally, if all of the words refer to the same node C_i, then that is the concept desired by the user. However, multiple possibilities arise when the nodes are not all identical. The user may be interested in all of these concepts, their intersection, or some other subset. Similar ambiguities arise even in the absence of a conceptual structure like Wordnet. We may imagine the user to be interested in words similar to all of the words in the seed set, or to some subset of them. To disambiguate, we provide two logical operators for combining words in a seed set. Using the OR operator, the user indicates that the dictionary should contain words that are similar to *at least one word* in the seed set. Alternatively, using the AND operator, he can indicate that words in the dictionary need to be similar to *all words* in the seed set. In general, it is possible to accommodate all combinations of logical operations between seed words, but so far these two operators have been expressive enough in practice.

The algorithm in Figure 1 can be extended in two different ways for seed words. The first possibility is to construct an aggregated context vector (depending on the operator) for the seed set from the context vectors of the individual seed words, and then use the aggregated context vector to identify candidate words in steps 2-4. The problem with this approach is that it lacks transparency from the users point of view. Since the user did not directly provide the aggregated context vector it is difficult to *explain* the returned ranking. The alternative that we use is shown in Figure 2. Candidates are identified using contexts of all seed words (steps 2-5). The similarities for candidate words are then computed with each seed word w_i^s and aggregated according to the operator (step 9). For OR, we take the maximum, and for AND the minimum of the individual similarities. The aggregated similarity scores are again returned in a sorted order.

One difficulty arises with using seed sets for dictionary

```
Algo BuildDict(Seed Set S, Int k, Double t, Operator +)

1.  Initialize candidate set CS to empty set
2.  For each word w_i^s ∈ S
3.      For all words in WV(w_i^s)
4.          If WV(w_i^s, w') > some threshold t'
5.          Add w' to CS

6.  For each word w' in CS
7.      For each word w_i^s in S
8.          Compute similarity sim(w_i^s, w')
9.      Aggregate similarity for w' using operator +
10.     Reject w' if similarity below threshold t
11. Sort remaining words by similarity and return top k
```

Figure 2: Dictionary building algorithm with a Seed Set

construction. Deciding whether or not to include a returned word becomes harder for the user, since he may be unsure which seed word resulted in its inclusion in the ranking. To help the user in this task, along with each returned word w, we provide as evidence the sorted list of top seed words (optionally with the relevant context words) to which w is similar.

An interesting situation arises when the user specifies the AND operator indicating that he is interested in words similar to *all* seed words, but very few (significantly less than parameter k) words from the vocabulary satisfy this condition. In Sec. 5, we will see how this issue becomes relevant when re-using dictionaries. Of course, one possibility is to simply return an empty ranking. But it is not very helpful for the user in constructing this dictionary, since he now has to try out different subsets of the seed set. A better solution for the user is to partition (or cluster) the seed set into sub-parts with a coherent sense, and then return the parts and their corresponding ranked list of words for AND as the operator.

This clustering of the seed set is done inside the algorithm in Figure 2 by reducing it to the task of finding connected components in a graph, as follows. After computing the similarities of the candidate words with seed words in step 8, we construct an undirected bipartite graph between the candidate words and the seed words by adding a 'similarity' edge (w', w_i^s) only if the similarity $\text{sim}(w_i^s, w')$ is above a certain threshold. Instead of creating a new parameter for this, we re-use the user specified minimum similarity parameter t, since it has a similar interpretation. Then we find the connected components in the bipartite graph. Each connected component now consists of related seed words and candidate synonyms. So, for each connected component, we return the seed words in it as a suggested sub-dictionary along with the ranked list of similarities of the candidate words in the same component. In effect, we split a seed set into coherent subsets such that each sub-set can seed a new dictionary.

4.3 Positive and Negative Seed Sets

In the third supervision model, in addition to providing examples of words that he wants in the dictionary, the user can also provide examples of what he does not wish to be included. This can significantly enhance the expressibility of the user's language when providing supervision. For example, the user can provide 'agent, rep, representative' as the positive seed set for constructing a 'Contact Center Agent' dictionary, and he can additionally include 'manager, mgr, supervisor' in the negative seed set to indicate that he does not want words referring to supervisors to be included in the dictionary. Eliminating these words and their close synonyms can significantly increase the recall of the top k ranking.

Specifically, in this model, the user provides two seed sets, P^s and N^s. As before, we use logical operators to define the semantics combining the two seed sets. In the case of negative examples, typically the user wants to leave out words that are similar to *any* of the negative seed words. Accordingly, we interpret the combination of a positive and a negative seed set as $+\{w_i^p \in P_s\}$AND NOT$\{w_i^n \in N_s\}$, where the operator + may be AND or OR. While it possible to provide the flexibility of arbitrary combinations of operators, we have found these two interpretations to be expressive enough for our use cases so far.

To handle positive and negative seed sets, we extend our algorithm in Figure 2 and call it BuildDict$(S^p, S^n, k, t, +)$. The only difference appears in Step 9, where we compute the aggregate similarity by taking into account the negative seed words as well. Recall that for any candidate word w, we aggregate its similarity over the positive seed words using maximum or minimum depending on the operator +. The natural way to accommodate the negative seed words is to subtract from this aggregate score the maximum similarity sim^n over all negative seed words. However, this often leads to sharp decrease in overall similarity scores. So, we use of an exponential decay model using sim^n, so that only words with very high similarity with any negative seed word are affected.

4.4 Interactive Supervision

The supervision models discussed so far are all 'offline' in nature, in that the entire set of positive and/or negative seed words need to be provided to the dictionary building algorithm in advance. This is often difficult to do in practice when exploring a new dataset. Typically, the user may not know the right positive and negative seed words in advance. For example, the user may become aware that the word 'rep' is used to refer to contact center agents only after observing it in the initial ranked list, and then he can include it in the positive seed set. Thus an 'online' interactive framework is more natural, where the user starts off with a small set of words, inspects the results, selects and rejects words from the returned ranking, and iterates until he is satisfied. His interactive supervision then provides our algorithm with the positive and negative seed words at each stage of the iteration, and the seed sets gradually become refined and the ranking comes closer to the user's preference as the iterations continue.

The overall interaction framework is shown at a high level in Figure 3. Observe that the user can also modify the minimum similarity threshold, the desired number of synonyms and the combination operation interactively as he thinks fit.

4.5 Multi Dictionary Construction

Typically, a user needs to build several dictionary-based annotators for exploring and analyzing any dataset. The naive approach for doing this is to construct each of n dictionaries D_1 to D_n independently and sequentially using the supervision models discussed earlier in this section. However,

```
Algo InteractiveBuildDict(Seed Set S)

1.  Initialize S^p to S and S^n to empty set
2.  Initialize length k, thresh t, operator +

3.  Invoke BuildDict(S^p,S^n,k,t,+) of Sec. 4.3 for ranking R
4.  While user is not satisfied with R
5.      Refine S^p and S^n from feedback
6.      Get optional feedback on k, t, +
5.      Rebuild R using BuildDict(S^p,S^n,k,t,+) of Sec. 4.3
```

Figure 3: The Interactive Dictionary Building Framework

this procedure can be made significantly more accurate and efficient by constructing them simultaneously. Ideally, the set of annotators written for any dataset should not overlap significantly, so their dictionaries should also not have too many shared words. Thus constructing one dictionary can benefit significantly from the knowledge of other dictionaries that are of interest for the same collection of documents. For example, when constructing two dictionaries for 'Car Parts' and 'Service Center' simultaneously, it may be clear that 'service' should be included in the second dictionary and not the first. But it is not as clear when constructing only the 'Car Parts' dictionary individually since car parts usually get used or repaired in service centers.

In the multi-dictionary construction approach, the user provides n seed sets (or pairs of positive and negative seed sets) to the algorithm and gets back n different rankings, one corresponding to each (or each pair) of the seed sets. The additional knowledge available to the algorithm is that the same output word should not appear in more than one returned ranking. The algorithm in Figure 2 can be extended to handle n seed sets by assigning each candidate word to *only* its most similar seed set and not any of the others. We will motivate and discuss applications of multi-dictionary construction in the context of reusing sets of dictionaries in Sec. 5.

4.6 Seeding using Context

In all of the models above, the user provides words as example of what he wants or does not want to be included in the dictionary. The algorithms for dictionary construction then construct the context vectors of these words to interpret what they 'mean' in the context of the document collection, and then proceeds to find words that are similar (or different) in meaning. One short-coming of this supervision model is that the algorithm cannot proceed when given out-of-vocabulary words as initial seeds. For example, when 'agent' is provided as the seed word, no similar words can be found for a document collection where 'agent' is never used and the appropriate word is 'representative' instead. In an alternative model, we may imagine the user as directly providing the 'meaning' of the words by specifying a context vector. For example, the context vector can include words referring to tasks performed by an agent in a contact center. All of the algorithms above can then directly start from this context vector without having to construct it and then proceed as before. It is also possible to imagine models where the supervisor provides both seed sets and context vectors and the algorithm combines the two appropriately for constructing the initial context vector. While apparently this context specification model does not seem as natural as the supervision-by-seeding model, in Sec. 5 we will see scenarios in the context of dictionary re-use where it becomes very useful.

5. RE-USING DICTIONARIES

As motivated in Sec. 2.2, a team of one or more practitioners engaged in providing text analytics services have to annotate and analyze several document collections from similar and related domains across different engagements. They need to construct similar annotators, and as a result similar dictionaries, for many different document collections. Thus assetizing and re-using dictionaries is profitable from multiple angles, such as efficiency, standardization etc.

In this section, we will consider a scenario where a repository of dictionaries, constructed from and consolidated over multiple datasets from different domains, is available to the practitioner. Automatically organizing and maintaining repositories as a consolidated structure presents technical challenges of its own. For the purposes of this paper, we will assume that such a repository is available — possibly constructed and maintained manually from dictionaries created over different datasets using algorithms outlined in Sec. 4. In general, the repository R could be a complex hierarchical structure such as WordNet, where each node corresponds to a concept and has its associated dictionary, and concepts occurring closer to the root represent more general concepts. We discuss one such repository that we have constructed and maintain in Sec. 6.

Our biggest challenge in Sec. 4 for constructing a dictionary for a concept was around describing or specifying a semantic concept in the absence of a semantic structure. Once the user has a repository R at his disposal, this task becomes significantly easier — he may simply refer to a node $n \in R$. Each node n already has its own dictionary of words $n.D$. However, note that the same dictionary cannot directly be used for a new dataset, which may be from a new domain and involve a different vocabulary. Therefore a concept node N needs to be 'adapted' for a new document collection. This involves throwing away of out-of-vocabulary words and words that are used in a different sense, and including new words currently not in the dictionary. So the dictionaries still need to be constructed for every new dataset, and the existing concept nodes can be used for seeding.

We now look at how such a repository helps in specifying concepts and building new dictionaries for a new document collection using algorithms in Sec. 4. The dictionary $n.D$ associated with a concept node can directly be used as a positive seed set for the algorithm in Sec. 4.2. The ranking that is returned is inspected by the user to create the adapted dictionary adapt($n.D$) for the new document collection. One issue that assumes greater significance in the context of dictionary adaptation is the possible lack of coherence among the old dictionary words for the new dataset. The ability to split one dictionary into multiple ones to ensure semantic consistency, as described in Sec. 4.2, becomes important in such cases.

Given a new document collection, a practitioner typically needs to build several dictionaries for it. The concept nodes for many of these dictionaries may already be present in the repository. In such a scenario, instead of adapting the existing relevant dictionaries one at a time, the practitioner may simply select and choose all relevant nodes from the reposi-

tory at one shot and adapt them using the multi-dictionary builder algorithm in Sec. 4.5. In addition to saving the pratitioner precious time, we will see in our experimental evaluation (Sec. 6) how the accuracy of dictionary construction is actually improved as well.

One problematic scenario that commonly arises when adapting dictionaries for new datasets is that, in spite of a concept node n being relevant, none of the words in the concept dictionary $n.D$ occur in the vocabulary of the new dataset. In such situations, seeding using context (Sec. 4.6) becomes useful. In addition to the concept dictionary $n.D$, a repository node n can also store the context vector $n.W$ associated with the concept. Now, when all (or a large fraction) of the dictionary words $n.D$ are detected as out-of-vocabulary for the new dataset, dictionary construction algorithms can use both the seed dictionary $n.D$ and context $n.W$ for adapting a repository concept for the dataset.

In the next section, we discuss experimental evaluation of our proposed approaches for dictionary construction and adaptation in real analytics engagements across domains.

6. EXPERIMENTS

Our goal behind proposing this dictionary-building and reuse framework was enabling practitioners make best use of their knowledge and effort from previous text mining tasks. The success of the framework is measured from time savings (of about 60%) and subjective judgments of the practitioners. However we are also interested in concretely measuring benefits in terms of accepted metrics like precision and recall – but extensive experimentation proved to be difficult due to lack of public tagged corpora. In this section we present some experiments and case-studies that show the efficacy of the proposed methods. We first describe the real-world datasets and tasks where we applied our methods. We then present dictionary construction examples, and experiments demonstrating effectiveness of user interaction and dictionary re-use. The real proof of this framework lies in the real-world industrial best practices that can emerge as a result of these ideas, and we present our experiences in Sec. 7.

6.1 Datasets and Tasks

CRM Analytics. We used several CRM voice of customer (VOC) datasets in the form of customer satisfaction (CSat) survey forms, from electronics(15,000 surveys), telecom(20,000), and automobile(6,000) company helpdesks. The dictionaries constructed are used to find reasons for dissatisfied customers, and subsequently implement operational improvements in call centers. This is usually done by correlation reporting or predictive modeling on top of annotated records using concept dictionaries as building blocks [1]. Some common dictionaries in CRM VOC analysis are mentions of agent accent, communication issues, knowledge levels, agent authority issues, particular product and service problems, competitor comparisons, operational issues like hold time, accessibility, etc. Please see Sec. 7 for more details.

Other domains. We have applied our dictionary construction and adaptation techniques in a variety of other domains like social media mining and data cleansing. We analyzed Twitter feeds and built dictionaries for finding trends and topics frequently discussed by people and for finding intents, product reviews, and activities. We also adapted sentiment analysis dictionaries – 'high' may be a typical positive sentiment word but not when it refers to credit card interest rates. We also used our dictionary construction methods to help build data cleansing models for unstructured address datasets. We only report results with CRM analytics here.

6.2 Interactive dictionary creation

We first consider an example of constructing target concept dictionaries, starting from seed words. We denote positive feedback from the user using a + and negative responses with a − superscript. Imagine an automobile domain, where we want to capture the various mentions of car noises in customer surveys or agent notes. Such a dictionary can result in product defect insights during aggregate analysis. Starting with 'noise' as a seed word, the top 5 synonyms returned by our algorithm along with perceived relevance are: $sound^+$, $vibration^-$, $brakes^-$, $rattling^+$, $problem^-$. On incorporating this positive and negative feedback, the new additional synonyms are: $grinding^+$, $popping^+$, $rumbling^+$, $clunking^+$, $whining^+$. This example uses OR semantics to search for similar meaning words and is illustrative of the power of interaction and feedback. A good noise problems dictionary (and resultant annotator) can now be created iteratively; we have illustrated just one feedback iteration here. Next, we would like to measure the actual effectiveness of dictionaries constructed by this interactive process.

Ideally, we would like to measure the goodness of the dictionaries constructed compared to gold-standard dictionaries. Unfortunately, such gold standards are hard to come by in industrial settings. An alternative is to measure the goodness of these dictionaries in downstream text processing tasks such as annotating documents. We measured precision and recall of annotating survey documents with target concepts. We hand tagged a set of 630 survey documents from the automobile domain with target concepts of 'pleasant agent', 'timeliness', and 'car parts'. Inter-human annotator agreement for this tagging was not explicitly measured, but we expect about $20\% - 30\%$ disagreement due to the subjective nature of the concepts involved from earlier experience[5]. We constructed dictionary-based concept annotators which annotated mentions of the dictionary words.

We interactively construct timeliness, car parts, and pleasant agent dictionaries and measure precision and recall (and F1) as the dictionaries evolve. For lack of space, we report results after just one iteration and restrict dictionary size to 10. Table 1 records the target concept, the seed word, aggregation semantics, and top 10 synonyms for 2 rounds of synonym finding – before and after user feedback. The last column shows a loose upper bound $Recall_{max}$, as the maximum achievable recall by the 'best' 10 word dictionary. $Recall_{max}$ is directly computed from the tagged corpus by counting occurrences and measuring recall for a dictionary of the 10 most frequent tagged words. The 'pleasant agent' dictionary is very precise to start with and loses a slight amount of precision for significant gains in recall even for just 10 words. Note the final recall of 51% which is very close to the maximum possible 57%. 'Timeliness' is a difficult subjective concept to tag manually. Yet we achieved nearly 60% precision at 16% recall – maximum possible recall was also low at 33% showing the subjectivity and 'spread' of the concept. The 'car parts' dictionary was constructed with over 63% precision and merely 5% recall as against a max-

Concept	Iter#	OR/AND	Synonym list with feedback	*Precision*	*Recall*	F1	$Recall_{max}$
Pleasant agent (seed: *kind*)	1	OR	$polite^+, helpful^+, friendly^+, professional^+, courteous^+, nice^+, pleasant^+, discouraged^-, aspects^-$	91.4%	43.2%	58.6%	57.6%
	2	OR	$polite^+, helpful^+, friendly^+, professional^+, courteous^+, nice^+, pleasant^+, very^+, good^+$	80.1%	51%	62.3%	
Timeliness (seed: *minutes*)	1	OR	$forever^+, hung^+, minute^+, while^+, years^+, finished^-, 20^-, 15^-, 16^-$	57.1%	6.3%	11.3%	33.8%
	2	OR	$forever^+, hung^+, minute^+, while^+, years^+, atleast^+, specifically^-, clock^-, wait^+$	59.5%	16.6%	25.9%	
Car parts (seed: *door*)	1	AND	$opener^+, locks^+, mat^+, latch^+, trunk^+, automatic^+, garage^-, 60^-, open^-$	43.7%	1.6%	3.1%	40.2%
	2	AND	$opener^+, locks^+, mat^+, latch^+, trunk^+, automatic^+, side^+, home^-, motor^+$	63.4%	5.2%	9.6%	

Table 1: Utility of interactive dictionary construction

imum possible 40%. The word 'parts' for this dictionary was responsible for over 7% recall, and the algorithm happened to miss this word. Across the board, we observe that F1 improves significantly with just one iteration. In reality, the interactive dictionary building process is iterative and manually tuning is always possible. Dictionaries also contain many more than just 10 entries, but we have shown the effectiveness of feedback even for such small dictionaries in this experiment.

Dictionary	Precision	Recall
Electronics 'Pleasant agent' → Automobile 'Pleasant agent'		
kind, thorough, courteous, helpful, polite	91.9%	29.3%
kind, thorough, courteous, helpful, polite, friendly, nice, professional, knowledgeable, pleasant	87.7%	46.6%
Telecom 'Timeliness' → Automobile 'Timeliness'		
minutes, hours, wait, while, mins,	60.1%	15.2%
minutes, hours, wait, while, mins, forever, hung, putting, hold, research	49.6%	24.2%
Fictional 'Car parts' → Automobile 'Car parts'		
door, window, steering, tire	90.6%	13.3%
door, window, steering, tire, motor, garage, informed, open, passenger, flat	75.7%	16.1%

Table 2: Adapting dictionaries

6.3 Adapting dictionaries

Next, in Table 2 we turn to adapting dictionaries from other domains to automobiles, and measure precision and recall on our tagged corpus as earlier. We would have tried in-domain adaptation if we had other automobile dictionaries available. Again, we take precision and recall of resultant dictionary-based annotations as a proxy for measuring the goodness of the adaptation for the 3 target dictionaries. To begin with, we use a 'pleasant agent qualities' dictionary from an electronics company's helpdesk data, a 'timeliness issues' dictionary from a financial company's call center data. For 'car parts' we did not have access to a separate dataset and created a fictional 'car parts' dictionary. In Table 1, for each of these dictionaries we show their precision and recall if used as-is in the respective first rows. Next, with *OR* semantics, we adapt them to the target automobiles dataset, and present the adapted dictionaries with their precision and recall on the tagged corpus of 630 surveys. The 'pleasant agent' dictionary from electronics is applicable as-is with very good precision of 91% but low recall of 29%. Recall improves to 46% on adaptation as against $Recall_{max}$ of 57%. Similarly, the finance 'timeliness' dictionary is applicable as-is with moderate precision and low recall. However, it achieves a final precision of 49% and recall of 24%. The recall improvement is significant in the light of a $Recall_{max}$ of 33%. 'Car parts' being a much larger target dictionary achieves good final precision but lower recall. Perhaps, it would have been helped by a real related dictionary of another automobile dataset to adapt from. F1 values also always improves, but are not shown for space constraints. This performance shows that this is a very good starting point for dictionary construction in a new domain which the practitioner may not be too familiar with. It is clear that these numbers will go down when trying to adapt more domain specific dictionaries, and we will see this next.

In the above experiment, we adapted the 3 dictionaries simultaneously using the multi-dictionary presented in Sec. 5. Independent adaptation led to very similar final numbers, not reported here for space reasons. This is not surprising give the very distinct senses of these three concepts. We then conducted a small experiment to really test the efficacy of multi-dictionary adaptation. We did multi-dictionary adaptation for two dictionaries: 1) a 'pleasant agent' dictionary: *kind*, *nice*, *pleasant*, *polite*, and 2) a 'agent knowledge' dictionary: *knowledgeable*, *thorough*, *efficient*, *investigated*. Note that in all previous experiments, we were satisfied with a fairly general sense of 'pleas-

ant agent'. However, now when this has to compete with 'agent knowledge', words such as 'efficient' and 'knowledgeable' no longer belong to 'pleasant agent'. This is precisely the purpose of multi-dictionary adaptation. All these words may belong together in a more abstract 'agent qualities' dictionary, but finer discrimination is now needed. The result of this adaptation was a new 'pleasant agent' dictionary: *kind*, *nice*, *pleasant*, *polite*, *helpful*, *friendly*, *courteous*, *cordial*, *informative*, *considerate* with precision 89.1% (higher than the adapted one in Table 2). The resultant 'agent knowledge' dictionary was: *knowledgeable*, *thorough*, *efficient*, *investigated*, *slow*, *enjoyed*, *cooperative*, *responses*. While we have no tagged data to measure performance, the distinct sense of this dictionary is obviously to capture mentions of agents having and applying knowledge to solve complex problems.

Discussion. Some additional observation can be made from Table 1 and Table 2. Interactively building dictionaries from scratch leads to good dictionaries, but adapting earlier dictionaries also leads to dictionaries of similar quality. Interactively building dictionaries led to better precision for 'pleasant agent' and 'timeliness' than adapting them. Adaptation on the other hand led to much better recall for 'timeliness' and 'car parts', where the concepts at hand are very broad and subjective. Note the significant amount of time can be saved by practitioners by choosing the right dictionaries to adapt, and then apply their domain knowledge to decide which dictionaries to improve further interactively. Estimates from real practitioners suggest that adapting dictionaries consistently results in 50–60% time savings, compared to starting new dictionary building tasks from scratch. We are looking to do further user studies to validate this.

All these methods sit together in the arsenal of our text mining practitioners, and adaptation is usually followed up by interactively improving dictionaries always guided by intuition and domain knowledge. This is in contrast to the current state of the art where products ship with pre-packaged components, but further construction and maintenance of dictionaries are left as a manual task for the practitioners. These experiments and examples clearly demonstrate that the right set of dictionary creation and adaptation tools can equip practitioners to efficiently perform dictionary-based text mining tasks in a wide variety of domains. We recount our experiences and the emergence of a consistent methodology around this in the next section.

7. EXPERIENCES AND BEST PRACTICES

We have seen dictionary-based text mining applied to diverse areas, from CRM analytics for automotive, telecom, retail, to social media mining and legal applications. An emerging trend in the text analytics industry is to move from product-centric to service-centric offerings. This saves software costs and offers the expertise of practitioners (data miners or business analysts) to clients who need not worry about how the technology works. Such a commoditization of text mining necessitates a level of tool and process standardization for the practitioners which is lacking as of now. Note that pre-packaged components in commercial frameworks follow a part of this principle and ship with dictionaries per industry (telecom, finance, healthcare) and intended usage type (customer service analysis, social media mining, online review/blog/forum analysis). While this is a start, we believe the evolution, customization, and re-use of these components and linguistic resources are what will lead to more effective practitioners.

As researchers, we have worked on designing and developing commercial text analytics service offerings in the CRM area[1]. This enabled us to work together with practitioners and observe the potential for commoditization in dictionary-based text mining. The work presented in this paper resulted directly from this interaction with practitioners, and also allowed us to dwell upon questions like long-term re-use and process standardization. In engagements, practitioners will use tools and products in an ad-hoc manner to solve their current problem. The prevalence of human intervention and supervision in real-life dictionary-based text mining presented us the opportunity to make tools that facilitated interactive dictionary building, adaptation, and re-use.

We have already seen the time and effort savings and accuracy improvements these tools can bring about in our experiments. We also saw the supervision required in higher order decisions like '*which set of dictionaries to build?*' for the dataset at hand. While we ignored this question while talking about interactively building dictionaries, it is central to applying transfer learning and adaptation ideas across domains. We now fit the various pieces discussed in this paper together in a proposed methodology for asset-based and standardized dictionary-based text mining. We envisage an evolving repository of dictionaries, possibly organized as a hierarchy, that will be a crucial tool in the arsenal of text mining practitioners. Figure 4 shows one specific real-world instantiation of such a repository that is built from scratch and has evolved over several CRM analytics engagements (some industry specific dictionaries omitted). Such a repository has proved a great help as seen in our experiments and examples of one shot multi-dictionary adaptation. This repository along with the other dictionary construction tooling is a concrete asset that takes a step toward re-use and standardization that is important to analytics services.

For the area of CRM analytics, the idea in the example repository is to have a clear set of domain independent dictionaries (say customer service or contact center issues) that can be quickly adapted to new datasets with no or very little human input. We have seen the efficiency and time savings of this in Sec. 6.3. At the same time it is important to build a set of domain specific dictionaries over time that can at least be adapted in datasets of the same industry. It is clear how a text analytics engagement for say a telecom company A can be speeded up significantly is the human knowledge that has gone into building dictionaries for telecom company B can be effectively re-used. Across domains too, had we had a 'bike parts' dictionary or 'car parts' dictionary of another company, our seeding and subsequent results for 'car parts' would have been better than the fictional seeds we used in our experiments in Sec. 6. In telecom as well as electronics we did have 'phone parts' and 'computer parts' dictionaries but the specific terms did not even occur in the automobile domain. In Sec. 4.6 we showed how dictionaries can be adapted from contexts even without seed sets. If we had a repository like the example one containing 'computer parts' with its context of complaints, breakages, and fixes, we could adapt it to build a 'car parts' dictionary.

Such *asset-based usage* is an attractive next step in the evolution of text mining as it moves on from using pre-packaged components to adapting, customizing, and enrich-

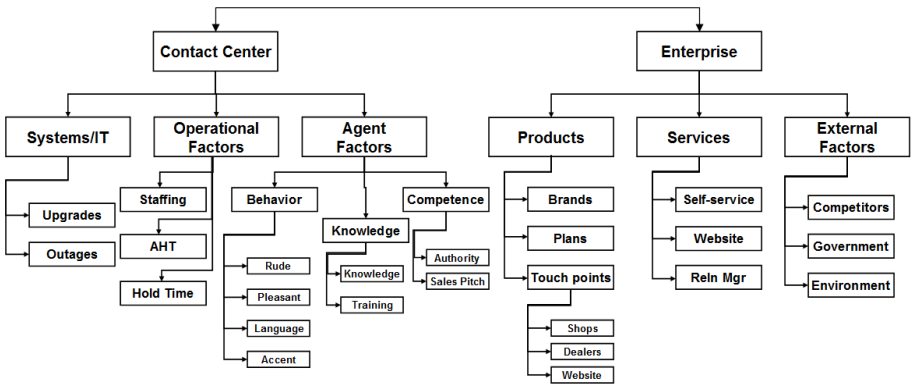

Figure 4: An example evolving dictionary repository

ing back these components. In effect, we have begun to see repeatability and time/effort savings for practitioners by following our proposed methodology of creating and re-using private domain-specific repositories. We believe this is a potential standard methodology for practitioners – to define and semi-automatically create private domain-specific Wordnets in the form of a repository of dictionaries and their contexts that can help dictionary-based text mining. While these are first steps, feedback from practitioners working in real-life engagements leads us to believe we are on the right track. We would like to see completely customizable off-the-shelf repositories become available for re-use across tasks and practitioners, thereby saving time, effort, and costs.

8. CONCLUSION

We presented interactive, corpus-aware dictionary construction, adaptation, and re-use techniques. We showed these techniques work well practically on real-world datasets. Due to our association with designing and building commercial text mining service offerings, we were able to dwell upon the important questions of assetization and re-use in text mining. We proposed an evolving repository of dictionaries with associated tooling that can significantly help reduce the time and effort practitioners spend on new dictionary-based text mining tasks. The main challenge facing the text analytics industry is scarcity of practitioners that understand technology and have business insights. Asset-based tooling we presented helps tackle this skills shortage by allowing knowledge re-use. While there is more to be done, we hope these are first steps toward standardization that helps the text mining industry mature with wide-spread deployments.

9. REFERENCES

[1] I. Bhattacharya, S. Godbole, A. Gupta, A. Verma, J. Achtermann, and K. English. Enabling analysts in managed services for CRM analytics. In *Proc. of KDD*, 2009.

[2] I. Bhattacharya, S. Godbole, S. Joshi, and A. Verma. Cross-guided clustering: Transfer of relevant supervision across domains for improved clustering. In *Proc. of ICDM*, 2009.

[3] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, 2007.

[4] R. C. Bunescu. *Learning for Information Extraction: From Named Entity Recognition and Disambiguation To Relation Extraction*. PhD thesis, Department of Computer Sciences, UTexas at Austin, 2007.

[5] S. Godbole and S. Roy. Text classification business intelligence and interactivity: Automating c-sat analysis for services industry. In *Proc. of KDD*, 2008.

[6] R. Jones, A. McCallum, K. Nigam, and E. Riloff. Bootstrapping for text learning tasks. In *IJCAI Workshop on Text Mining*, 1999.

[7] D. D. Lewis, R. Ghani, D. Mladenic, I. Moulinier, and M. Wasson. Workshop on operational text classification. In conjunction with KDD 2003.

[8] T. Li, V. Sindhwani, C. Ding, and Y. Zhang. Knowledge transformation for cross-domain sentiment classification. In *Proc. of SIGIR*, 2009.

[9] D. Lin. Automatic retrieval and clustering of similar words. In *Proc. of COLING-ACL*, 1998.

[10] L. Mihalkova and R. Mooney. Transfer learning from minimal target data by mapping across relational domains. In *Proc. of IJCAI*, 2009.

[11] S. O'Dowd. Unstructured data and text analytics in capital markets, IDC research document, doc fin212553/july, 2008.

[12] P. Pantel and D. Lin. Discovering word senses from text. In *Proc. of KDD*, 2002.

[13] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proc. of AAAI*, 1999.

[14] H. Takeuchi, L. V. Subramaniam, T. Nasukawa, and S. Roy. Getting insights from the voices of customers: Conversation mining at a contact center. *Inf. Sci.*, 179(11):1584–1591, 2009.

[15] S. Thrun and L. Pratt. *Learning to learn*. Kluwer Academic Publishers, Norwell MA USA, 1998.

[16] D. Wenyuan, X. Gui-Rong, Y. Qiang, and Y. Yong. Transferring naive Bayes classifiers for text classification. In *AAAI*, 2007.

[17] P. Wu and T. G. Dietterich. Improving SVM accuracy by training on auxiliary data sources. In *Proc. of ICML*, 2004.