Short Text Matching in Performance Management

Manoj Apte
Tata Consultancy Services
manoj.apte@tcs.com

Sachin Pawar[*]
Tata Consultancy Services
sachin7.p@tcs.com

Sangameshwar Patil[†]
Tata Consultancy Services
sangameshwar.patil@tcs.com

Sriram Baskaran
Tata Consultancy Services
sriram.baskaran@tcs.com

Apoorv Shrivastava
Tata Consultancy Services
apoorv.shrivastava@tcs.com

Girish K. Palshikar
Tata Consultancy Services
gk.palshikar@tcs.com

ABSTRACT

In "Role" based Performance Appraisal process the evaluation of Individuals is done based on the meeting of target for "Goals" given to that individual in the specified time period. Standardization of goals with the help of a pre defined "template" is important for completeness and correctness of role definition and comparing two individuals. Since a goal is a short textual description of expected activity we pose this as a matching problem and explore two different approaches that use minimal human supervision. First approach is based on co-training framework which uses goal description and available additional information in the form of self comments. The second approach uses semantic similarity using weak supervision framework. We demonstrate superior performance of the two approaches as compared to multiple baselines. First approach gives better recall while second approach scores better in precision. Based on the objective of the end application any of the approaches can be used.

1. INTRODUCTION

Quality, utilization and productivity of the workforce are very important factors to be considered by Human Resource (HR) Department of an organization. Performance Appraisal (PA) of the workforce focusses on the Quality factor and is used for identifying Top Performers and laggards in the organization. As organizations become larger it becomes difficult for the HR to measure each individuals performance and so process oriented approach needs to be followed. One approach followed by large organizations in this direction is to move to Role based Performance Management. In Role based Performance Management, each person is mapped to certain Role based on the expected Responsibilities and Activities to be performed. The responsibilities and activities for each role have to be defined in such a way that they represent a large amount of tasks done by the workforce in the specified role. Once the Role, Responsibilities and Activities are identified it is important to set *Goals* with specified targets over a particular time period and then measure the performance based on the inputs given by the individual as well as the supervisor. Recently with the emergence of Analytic tools the PA process is also tracked for improvements based on process parameters. The results of this analysis also drive the process changes in the PA process.

We have worked on analyzing the PA process of a large Information Technology (IT) company employing more than $324,000$ associates and has a robust Performance Appraisal Process defined. The process involves the following:

1. Role is assigned for each person. A Goal template is associated with each Role. This goal template contains a list of standardized goals based on the role definition.
2. Supervisor sets the goals for each time period along with targets. This is known as Goal Setting Process. Here, the supervisor has to set at least 5 goals from the Goal Template for the corresponding role of the individual. These goals are known as *Template Goals*.
3. At the end of each time period, the individual writes the achievements and the remarks about the working environment from his/her perspective in *Self Comments* for each goal.
4. Based on the Self Comments, the Supervisor writes his/her opinion on the achievements of the individual and shortcomings if any.
5. The supervisor scores the individual based on the performance.

In the goal setting process, supervisors expect some freedom to assign goals which are not exactly fitting the Goal Template based on the activities the individual is expected to perform. As a result the supervisors are given freedom to set goals manually in addition to the template goals.

In our sample we have $156,904$ confirmed employees across 869 roles. The total number of Goals assigned is $2,176,974$ out of which template goals are $863,465$ (39.66%) and manually created goals are 1313509. The Role representing the largest number of individuals is 'Developer' ($46,476$ Individuals). The key statistics of the dataset of the Role 'Developer' are given in Table 1 below.

We find that supervisors have used their freedom to assign goals outside the template and assigned lot of manually

[*]Doctoral research scholar at Dept. of CSE, IIT Bombay
[†]Doctoral research scholar at Dept. of CSE, IIT Madras

#Individuals	46,476
#Goals	638,301
#Template Goals assigned	324,036
#Manually created goals assigned	314,265
#Manually created goals	46,853
Size of Goal Template	41

Table 1: Details for the role "Developer"

created goals. Some of these goals are likely to be very similar to one of the template goals. For example, following manually created goals are very similar to the template goal `Number of Certifications`:

1. `# Certifications`
2. `No. of certification taken`
3. `No of certifications obtained`
4. `Domain Certification`

High number of manually created goals poses the following problems :

1. Completeness and correctness of Role Definition - If unusually high number of manually created goals that are not similar to any of the template goal are assigned, then the Role definition should be reviewed to see whether some of the responsibilities and activities for the Role as expected by the supervisors should be added. Similarly,
2. Comparison of two individuals based on Goal assignment - Generally organizations follow the method of ranking the individuals and forced distribution where the individuals are compared with each other. In such a scenario, it is important that the expectations from the individuals are comparable. Such a comparison of expectations can be done by comparing the goals set for the individuals. It is not possible unless the goals are mapped against a standard.

To solve the above problems it is important to standardize individual collection of goals. It can be achieved by matching each manually created goal to its equivalent template goal. The manually created goals which are not equivalent to any of the template goals should be kept separate.

There are 3 types of inputs available which help in determining whether a goal description is matching with one of the template goals : (i) Template Goal description, (ii) Description of manually created goal and (iii) Self Comments written by individuals for the goal. For the matching of manually created goals with the template goals, we explore two different classification based approaches. The first approach uses a well-known co-training [3] framework and the second one uses semantic similarity using weak supervision [22] framework. The former approach uses all the 3 available inputs but no human-in-the-loop. The latter uses the goal descriptions along with human feedback and uses active learning to minimize human intervention.

The rest of the paper is organized as follows. Section 2 covers the related literature and its limitations for applying in our domain. Our co-training based approach is described section 3 and weak supervision based approach is described in section 4. We have compared our approaches with different baseline implementations and we present the detailed results in section 5. Finally, we conclude in section 6 with some discussion on future work.

2. RELATED WORK

In performance appraisal systems, generally supervisors set crisp, direct and objective goals. Therefore, most of the goal descriptions are quite short having average length of 10 words. Classification of short documents has been an area of active research in the natural language processing and information retrieval communities. Tweet classification is an increasingly important variant of the short text classification problem. Tweet classification to improve information filtering has been investigated in [26, 25]. They have focused on extracting tweet specific features (such as user call out, shortening of words, presence of time-event phrases, opinionated words etc.) and used standard supervised machine learning approach with naive Bayes algorithm as the learner. Another kind of short text is seen the textual survey responses. Classifying the short text documents seen in survey responses has been tackled by Giorgetti et al [11]. For this type of short documents, Giorgetti et al. observed that supervised learning methods outperform the dictionary based approach. Li and Yamanishi [14] use frequent patterns and association rule mining for classifying short text in automobile survey. Sun [27] proposed a simple, scalable and non-parametric approach for short text classification. It first selects representative query words using bag of words and Clarity score and then searches for a small set of labelled short texts best matching the query words. The predicted category label is the majority vote for the search results. Lu and Li [17] proposed a deep architecture for matching short texts for addressing various matching tasks like finding relevant answers to a given question. Other approaches for short text classification are proposed by Zelikovitz and Hirsh [30], Bobicev and Sokolova [4] and Chen et al. [6].

Due to scarcity of labelled data and high cost involved in construction of it, semi-supervised learning approaches are used. Zhu [31] surveys various semi-supervised learning approaches such as self-training, co-training, Expectation Maximization (EM) and different graph based methods. Nigam et al. [20] proposed an approach for semi-supervised text classification from labelled and unlabelled documents using EM. Co-training [3] is a popular semi-supervised approach for training a classifier using both labelled and unlabelled data. Co-training requires two mutually independent feature views where each individual view is sufficient for classification. Two models are trained with initial labelled data, each using one of the feature views. They are then used to classify the remaining unlabelled records and very high confidence predictions by either of the model, are added to the labelled data. The models are re-trained using the additional labelled data and the process is repeated. Short text with side information can be easily mapped to two separate feature views - i) Features derived from short text itself; and ii) Features derived from the side information. Hence, co-training is one of the natural choices for addressing the problem of short text matching with side information. Nigam and Ghani [19] proposed a hybrid algorithm namely co-EM which is a combination of co-training and EM. Like co-training, it uses two feature views and like EM, it probabilistically labels all the unlabelled data. Another approach by Ghani [10] proposes a framework to incorporate unlabelled data in Error-Correcting Output Coding (ECOC) by decomposing multiclass problem into multiple binary problems and then using co-training to learn the individual binary classification problems. This was designed to scale up for multi-class classification with large number of classes.

There are special techniques designed for classification when labelled data is available only for positive class. Li and Liu [15] and Liu et al. [16] consider this problem of text classification with only one class of labelled documents and a set of unlabelled documents. Classifier is built in two steps. First step just identifies a reliable set of negative instances from the set of unlabelled instances. Second step iteratively builds a classifier using algorithms like EM. Fung et al. [9] and Elkan and Noto [7] also propose different approaches which use only positive data for learning.

3. METHOD 1: GOAL DESCRIPTION CLASSIFICATION USING CO-TRAINING

We model the problem of matching manually created goals to template goals as a classification problem. We opt for a semi-supervised learning based approach for two main reasons : i) Initial labelled data is limited in size and ii) unlabelled data is easily available.

Initial labelled data is constructed automatically by using the set of template goals where each template goal is assigned a distinct class label. Optionally, we can manually assign same class label to multiple template goals which are "semantically" similar. In addition to class labels covering the template goals, there is a special class label `NONE` which indicates that none of the template goals are matching.

We propose to use co-training framework [3] for semi-supervised learning. The major motivation for opting for co-training framework was that there is a natural separation of information used for classification of manually created goals. There are 2 different views of each goal (template goal as well as manually created goal):

1. V_1: Goal description itself
2. V_2: Self comments written for the goal

The intuition behind using V_2 is that similar goals are understood by the individuals in similar way and hence the corresponding self comments tend to be similar.

Two different classifiers are trained, each using features generated from only one of the views. We use Maximum Entropy Classifier with real-valued features.

3.1 Classifier C_1: Using Goal Descriptions

Classifier C_1 is trained using features generated only from the goal descriptions. For each goal description, various features are generated as follows:

1. Root-word of each word in the goal description becomes a feature. The value of the feature is set to δ^i where $0 < \delta < 1$ and i is the index of the corresponding word. The intuition is that value corresponding to a word feature is lower if that word appears later in the goal description. In practice, the value of δ equal to 0.95 is used.

2. If a goal description contains any two words such that both the words occur in a single "template" goal description, then such a combination of two words becomes a feature with a fixed weight of 1.5. Since the goal descriptions are generally short, we expect such pair of words to capture its essence in a better way.

3.2 Classifier C_2: Using Self Comments

Classifier C_2 is trained using features generated only from the self comments. For each goal description, various features are generated as follows:

1. A set of self comments is associated with each goal description. Root-words of all the words used in these self comments become features.

2. A bag-of-words is created for each goal description by using the set of associated self comments. This bag-of-words can be viewed as a large document. Following the Information Retrieval (IR) literature, TF-IDF (Term Frequency - Inverse Document Frequency) scores are computed for each word. For each word feature, its value is nothing but the corresponding TF-IDF score.

3.3 Challenge of Negative Instances

Our initial labelled data is constructed automatically using the set of template goals. This process does not label any goal with negative label, i.e. `NONE`. We cannot characterize the `NONE` class using a finite set of labelled examples. Hence, manually labelling examples of `NONE` class is not sufficient. This poses a challenge to train a classifier with just positive examples. We deal with this challenge in following way:

1. **Identifying candidate negative goals:** For each goal, we associate two word vectors which are vector-space representations using the views V_1 and V_2. For each manually created goal, we find its similarity with all the template goals. Both the views are considered and cosine similarity between the word vectors is used as a similarity measure. All those manually created goals having the highest similarity with any template goal, lower than a pre-defined threshold, are identified as "candidate negative" goals. Algorithm 1 describes the detailed procedure.

2. **Co-Training:** The "candidate" negative goals identified in the previous step are not considered as a part of unlabelled goals during the Co-Training iterations.

3. **Disagreement between two views:** After the termination of Co-Training process, the "candidate" negative goals are classified using both the classifiers. If two classifiers predict different class labels for any "candidate" negative goal and none of it is very confident about its classification, then such goals get the final class label as `NONE`.

3.4 Challenge of Class Imbalance

In each iteration of co-training, the unlabelled instances which are classified with high confidence are added to the labelled data along with the predicted label. As a template goal corresponds to a class label in our case, depending on the variation in the manually created goals belonging to a particular template goal, the confidence values assigned for

```
   Data: $T$ (Set of template goals), $N$ (Set of manually
         created goals), $\alpha$ (Weight given to $V_1$, lies
         between 0 and 1 with default value = 0.6), $\theta$
         (Similarity threshold on cosine similarity with
         default value = 0.2)
   Result: $N_{neg}$ (Set of manually created goals which are
           candidates for negative (NONE) class)
 1 $WV_1 := []$;   /* Empty mappings with $key = goal$ and
   $value =$ word vector using $V_1$ */
 2 $WV_2 := []$;   /* Empty mappings with $key = goal$ and
   $value =$ word vector using $V_2$ */
 3 foreach $g \in T \cup N$ do
 4     $WV_1[g] :=$ TF-IDF word vector using goal
       description of $g$;
 5     $WV_2[g] :=$ TF-IDF word vector using self comments
       for $g$;
 6 end
 7 foreach $g \in N$ do
 8     $S_{max} := 0$;
 9     foreach $g' \in T$ do
10         $sim_1 := CosineSim(WV_1[g], WV_1[g'])$;
11         $sim_2 := CosineSim(WV_2[g], WV_2[g'])$;
12         $sim := \alpha \cdot sim_1 + (1 - \alpha) \cdot sim_2$;
13         if $sim > S_{max}$ then $S_{max} := sim$
14     end
15     if $S_{max} < \theta$ then $N_{neg} := N_{neg} \cup g$
16 end
17 return $N_{neg}$;
```

Algorithm 1: Algorithm *GetNegativeCandidates* to determine candidates for negative (NONE) class

each class may vary. If the only criteria for adding instances to the labelled set is to check whether the classification probability is more than some threshold, then some class imbalance may get introduced in the labelled set in each iteration. Figures 1 and 2 show the skewed class distribution in the first iteration of co-training algorithm if a fixed threshold is used to select instances to add. In order to prevent this, our algorithm enforces a constraint on maximum and minimum number of instances of any class to be added to the labelled set in each iteration. This ensures that all classes get a fair representation in the labelled data.

3.5 Classification using Co-training framework

Algorithm 2 describes our co-training based approach in detail. The overview of this approach is as follows:

1. The initial labelled data is automatically created by labelling each template goals with different class label. All the manually created goals constitute the initial set of unlabelled goals.

2. A set of "candidate" negative instances is identified **(line 3)** and these instances do not participate in co-training iterations.

3. Two classifiers are trained using the labelled data : (i) C_1 using the goal descriptions view, V_1 and (ii) C_2 using self comments view, V_2 **(lines 6-7)**.

4. All the unlabelled goals which are not candidates for "negative" class are classified using both C_1 and C_2. The predictions along with classification confidence are recorded **(lines 9-17)**.

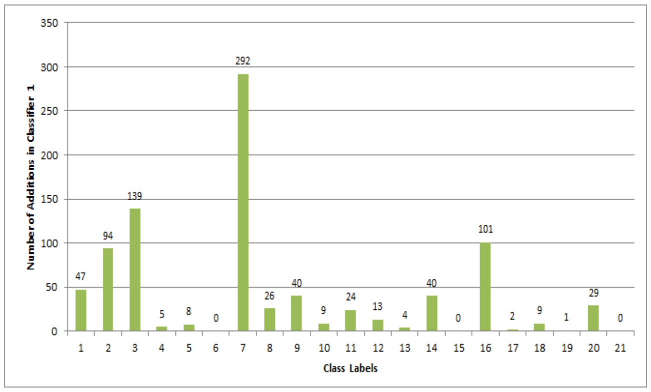

Figure 1: Class Distribution of instances classified by C1 with confidence more than some threshold in the first iteration

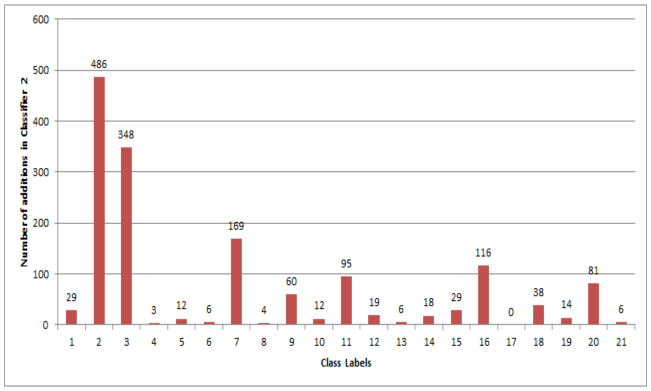

Figure 2: Class Distribution of instances classified by C2 with confidence more than some threshold in the first iteration

5. For each classifier, for each class label, unlabelled instances predicted with high confidence are added to the set of labelled instances such that the constraints on minimum and maximum number of additions are complied with **(lines 19-36)**.

6. The steps 3 to 5 are repeated till number of iterations reaches the specified limit or no new additions take place.

7. After co-training iterations, all the remaining unlabelled instances (including "candidate" negative instances) are classified using both the classifiers. If at least one classifier predicts a class label with the confidence greater than the defined thresholds, then the identified class label is assigned **(lines 41-46)**. Rest of the instances are labelled as NONE **(lines 47-48)**.

4. METHOD 2: GOAL DESCRIPTION CLASSIFICATION USING WEAK SUPERVISION

As discussed in earlier section, there are basically three types of inputs which help in determining whether a description of a manually created goal is similar to one of the available goals in the "template of goals" for a given role.

Data: T (Set of template goals), N (Set of manually created goals), I (Maximum number of co-training iterations), ADD_{max}, ADD_{min} (Maximum and minimum number of unlabelled goals per class in each iteration for each classifier), η_1, η_2 (Confidence thresholds for two classifiers), η_1', η_2' (Lenient confidence thresholds for two classifiers), η^a, η_1^f, η_2^f (Confidence thresholds used for final predictions)

Result: Set of tuples of the form (goal,label) where goal is from N and label is the matching template goal in T and `NONE` if none of the template goals match.

```
$L := T$ ;                                   /* Initialize to set of tuples (template goal, class label) */
$U := N$ ;                                   /* Initialize to set of manually created goals */
$N_{neg} := GetNegativeCandidates(T, N, 0.6, 0.2)$ ;                          /* $N_{neg} \subset N$ */
$iter := 0$;
while $iter < I$ do
    $C_1 :=$ MaxEnt Classifier trained using goal descriptions in $L$;
    $C_2 :=$ MaxEnt Classifier trained using self comments for each goal in $L$;
    $GC_1 := []$;$GC_2 := []$ ;   /* Empty two-level mappings with $key_1 = label$, $key_2 = goal$, $value = confidence$ */
    foreach $g \in U$ do
        if $g \in L$ or $g \in N_{neg}$ then continue;
        $(l_1, pr_1) :=$ Classify $g$ using $C_1$; $(l_2, pr_2) :=$ Classify $g$ using $C_2$;
        if $iter < J$ and $l_1 = l_2$ then   /* Agreement of classifiers is checked for first $J(< I)$ iterations */
            $GC_1[l_1][g] = pr_1$; $GC_2[l_2][g] = pr_2$;
        else if $iter \geq J$ then
            $GC_1[l_1][g] = pr_1$; $GC_2[l_2][g] = pr_2$;
        end
    end
    $N := 0$ ; /* Overall number of unlabelled goals added to $L$ in current iteration */
    foreach $label \in GC_1.keys$ do
        $N_{label} := 0$ ;       /* No. of unlabelled goals added by $C_1$ to $L$ with $label$ in current iteration */
        foreach $(g, pr) \in GC_1[label]$ do       /* selected in descending order of $pr$ for the given $label$ */
            if $(N_{label} < ADD_{max}$ and $pr > \eta_1)$ or $(N_{label} < ADD_{min}$ and $pr > (\eta_1' + (iter * 0.05)))$ then
                $L := L \cup (g, label)$; $N_{label} := N_{label} + 1$; $N := N + 1$;
            end
            if $N_{label} > ADD_{max}$ then break;
        end
    end
    foreach $label \in GC_2.keys$ do
        $N_{label} := 0$ ;       /* No. of unlabelled goals added by $C_2$ to $L$ with $label$ in current iteration */
        foreach $(g, pr) \in GC_2[label]$ do       /* selected in descending order of $pr$ for the given $label$ */
            if $(N_{label} < ADD_{max}$ and $pr > \eta_2)$ or $(N_{label} < ADD_{min}$ and $pr > (\eta_2' + (iter * 0.05)))$ then
                $L := L \cup (g, label)$; $N_{label} := N_{label} + 1$; $N := N + 1$;
            end
            if $N_{label} > ADD_{max}$ then break;
        end
    end
    if $N = 0$ then break;
end
foreach $g \in U$ do
    $(l_1, pr_1) :=$ Classify $g$ using $C_1$; $(l_2, pr_2) :=$ Classify $g$ using $C_2$;
    if $l_1 = l_2$ and $\frac{1}{2}(pr_2 + pr_2) > \eta^a$ then
        $L := L \cup (g, l_1)$;
    else if $pr_1 > \eta_1^f$ then
        $L := L \cup (g, l_1)$;
    else if $pr_2 > \eta_2^f$ then
        $L := L \cup (g, l_2)$;
    else
        $L := L \cup (g, NONE)$;
    end
end
return $L$;
```

Algorithm 2: Method 1 : Our Co-training based approach

The three types of inputs are: (i) the textual description of a goal in the set of "template goals" (i.e., the goals to be used as set of classes/categories in classification process), (ii) the textual description of a manually created goal whose similarity with one of the existing "template goals" is to be determined, and (iii) the text of comments (also called *self-comments* made by an appraisee in support of his/her achievements corresponding to the goal (i.e. the input goal in previous item (ii)).

Apart from the co-training based approach for classification mentioned in another section (which mainly uses the input goal description and self-comments as features), we propose to measure the similarity between the textual description of a "template goal" and the manually created goals to be classified. We note that often the goal descriptions are written more like phrase based text snippets and do not have complete sentence structure. In many ways the textual goal descriptions are similar to short text snippets observed in product review comments or textual responses to open-ended survey questions. Hence, we use the algorithm 3 which is a variant of the short text classification algorithms described in [22, 21] and adapt it to the current problem of manually created goal classification for the employee performance management problem. This algorithm uses weak supervision to minimize the human effort required to create labeled training data. We use a two stage iterative method in which we carry out first level classification using the similarity between the "template goal" description and the manually created goal without any human supervision (i.e., without use of labeled training data). The output of this stage is then passed through a weakly supervised learning stage. We use active learning paradigm for weak supervision to minimize the amount of feedback sought from human expert.

4.1 Stage I - Semantic text similarity based classification

We first compute semantic similarity between the "template goal" description and manually created goals to be classified. For this purpose, we represent each word/phrase in the textual description of "template goal" (i.e., class) using its WordNet [8] synset ids to capture the expected meaning of each word. Further, we assign a numeric weight to measure the relative importance of this word/phrase within the "template goal". For determining expected meaning of a word/phrase, we make use of unsupervised word sense disambiguation techniques [18, 12]. For a given word, this enables us to find out synonyms, antonyms as well as other related words (hypernyms, hyponyms etc.). To estimate relative importance of a word/phrase within the "template goal" description, we use the numeric weight assignment as described in [22]. We then find out word/phrase level overlap between the semantically enriched representation of "template goal" and the input textual description of manually created goal whose similarity with the "template goal" is to be computed. For the set of common words/phrases, the numeric weight signifying their relative importance is combined together. To compute the aggregate value of these possibly multiple relative importance values, we use the certainty factor algebra (CFA) [5]. If this aggregate value is above a pre-determined threshold, the manually created goal is deemed to be similar to the "template goal" and classified accordingly.

```
   Data: T (Set of template goals), N (Set of manually
         created goals), I (Maximum number of
         iterations)
   Result: Set of tuples of the form (goal, set of labels)
           where goal is from N and set of labels is the
           set of matching template goals in T and
           NONE if none of the template goals match
1  while $iter \leq I$ do
2      Stage-I:
3      foreach $t \in T$ do
4          $L_t = \emptyset$
5          Let $(w_1, w_2, \ldots, w_t)$ be the sequence of words in
           the textual goal description of template goal $t$.
6          Let $t' = (w'_1, w'_2, \ldots, w'_t)$ be new semantically
           enriched representation of $t$; where $w'_i$ = set of
           estimated word senses and corresponding related
           words (synonym, derivationally related words)
           of $w_i$ determined by using unsupervised WSD
           techniques [18] as well as considering the
           feedback received in the weak supervision stage.
7          foreach $n \in N$ do
8              $c_{nt'} = n \cap t'$
9              If the combined IDF-based relative
               importance of words in $c_{nt'}$ is above a
               threshold $\theta$, then $L_t = L_t \cup n$ (i.e., assign $t$
               to $n$ ).
10         end
11     end
12     Stage-II:
13     foreach $t \in T$ do
14         $C_t = Output\_of\_Clustering(L_t)$
15         foreach $c \in C_t$ do
16             Seek human feedback about correctness of
               assignment of $t$ to medoid of cluster $c$
17             Update $t'$ based on feedback.
18         end
19     end
20     iter++;
21 end
22 return $\{(t, L_t) | t \in T\}$
```

Algorithm 3: Method 2: Goal description classification using semantic similarity and weak supervision

4.2 Stage II - Weak Supervision using Active Learning

The classification carried out in the stage-I is vetted using human supervision. To minimize the human involvement and to use the weak supervision machine learning paradigm, we use active learning [24]. The most informative examples from the output of stage-I are selected using the active learning informativeness criteria. These examples are then presented to a human expert to ascertain whether the classification is correct. We cluster the manually created goals which have been deemed similar with a given "template goal" and then select a representative example for each cluster which is then queried to the human expert. To estimate the number of clusters, silhouette coefficient [28, 13, 23] is used. Silhouette Coefficient (ShC) is a practically useful measure to compare the trade-off between intra-cluster cohesiveness and inter-cluster separation. Silhouette coefficient for i^{th} data point is given by $ShC_i = \frac{b_i - a_i}{max(a_i, b_i)}$, where a_i is the average distance between i^{th} data point and other points in the same cluster; and b_i is average distance between i^{th} data point and all other points in the next nearest cluster. Silhouette Coefficient for a given clustering of data-points is average of individual ShC_i values. At run-time, multiple clusterings are tried out and the clustering having highest silhouette coefficient among the explored is chosen as the final clustering. A representative example from each cluster is chosen as the query to be posed to the human expert. The label verification by human expert is used for updated classification in the next iteration. The specific and detailed description of the technique is given in the algorithm pseudocode.

5. EXPERIMENTAL ANALYSIS

To demonstrate effectiveness of our approaches, we have compared it with classification using standard similarity measures like Cosine, Jaccard and Dice similarities with a defined threshold for each and off the shelf implementation of Nave Bayes and Maximum Entropy classifiers. Before we define the experiment setup we describe the datasets.

5.1 Dataset for Baseline Algorithms

We have used the performance appraisal dataset of an organization for one year. We have taken the goals that are set for the given year to all the associates. Since we are doing the analysis role-wise, we have identified the most frequently assigned role in the organization and run the different algorithms for that role.

The data for the algorithms contains original goal texts. We perform cleaning of the original goal text to get a cleaned goal text. The cleaning process starts with POS Tagging where each word of the text is tagged with the Part of Speech. We have used both Open NLP [1] and Stanford POS tagger [29] for POS Tagging. The choice of the POS Tagger didn't make a significant difference in the final classification results.

The goal names are very short and noisy text. They are bound to have spelling mistakes and human errors. We do a spell check of the goal names based on a corpus of valid words that are highly frequent in the given domain. We used Jazzy [2] for spell check and used the first suggestion (if present) as the valid word. Based on domain knowledge, we also replace known acronyms with their expansions. For example, all occurrences of `CSI` are replaced with `Client Satisfaction Index`.

The cleaning process removes all the punctuation marks and converts all upper case to lower case characters. It removes a set of stop words from the given goal name. Along with the general stop words like `and`, `the`, `on` etc., we also remove domain-specific stop words like `number`, `percentage`, `project`, etc. Based on domain knowledge, we replaced certain words in the goal text with their synonyms. For example, both the words `customer` and `client` (which are synonyms) are quite frequently used and we replace all occurrences of `customer` with `client`. We also performed lemmatization of the remaining words in the goal names to identify the root words based on their POS Tagging. For example, all the verbs like `training`, `trains`, `trained` are replaced with their root word `train`. Also, all the nouns like `trainings` and `training` are replaced with their root word `training`.

5.2 Dataset for Co-training based Approach

The dataset for the baseline algorithm will be used as training data for one of the classifiers. Apart from the goal names dataset created above, the co-training uses another view for the second classifier. The dataset for this view is taken from self comments written by the associates. For every goal we take a list of corresponding self comments and process them to form a TF-IDF based word vector.

5.3 Validation

We have performed all the experiments for the role "Developer". The Goal Template for this role contains 41 goals. We grouped "semantically" similar goals with the template so that they get the same class label. This resulted into 21 distinct class labels for the 41 template goals. For validation, we annotated 1000 manually created goals with one of the 21 newly defined class labels or `NONE` for the role "Developer". Henceforth in the paper we will call it as Gold Standard.

Our gold standard dataset consists of 1000 distinct goals where each goal can be assigned to multiple individuals. Total number of assignments for these 1000 distinct goals is $2,010,627$. From academic point of view, validation of classifiers is generally done on distinct records (goals). But in the industrial scenario, the validation in terms of number of assignments is more meaningful. Hence, we report the results on both distinct goals as well as total assignments.

We measure coverage [1], precision, recall, accuracy and f-measure for both the cases and compare them across different algorithms.

5.4 Evaluation Measures

Our classification algorithms predict one of the K positive classes or 1 negative class (`NONE`). The predictions of various algorithms on gold-standard dataset are compared with the manual annotations and a $(K+1) \times (K+1)$ confusion matrix C is defined as follows. The rows of this matrix correspond to the actual labels whereas the columns correspond to the predicted labels. Without loss of generality, we assume that the 0^{th} row and column correspond to the `NONE` label. Any cell C_{ij} of this matrix represents number of goals having i^{th} true label and j^{th} predicted label. Using this confusion matrix C, we compute following measures.

$$C = \begin{array}{cc} \textit{Predicted} & \\ \begin{bmatrix} c_{00} & \cdots & c_{0k} \\ \vdots & c_{ij} & \vdots \\ c_{k0} & \cdots & c_{kk} \end{bmatrix} & \textit{Actual} \end{array}$$

1. **True Positives (TP):** Number of goals having the predicted non-`NONE` label same as the gold-standard label.

$$TP = \sum_{i=1}^{K} C_{ii}$$

2. **False Positives (FP):** Number of goals having different gold-standard label than the predicted non-`NONE`

[1]Fraction of goals which are not classified as `NONE`

label.

$$FP = \sum_{j=1}^{K} C_{0j} + \sum_{i=1}^{K} \sum_{\substack{j=1 \\ j \neq i}}^{K} C_{ij}$$

3. **False Negatives (FN):** Number of goals having different predicted label than the non-`NONE` gold-standard label.

$$FN = \sum_{i=1}^{K} C_{i0} + \sum_{i=1}^{K} \sum_{\substack{j=1 \\ j \neq i}}^{K} C_{ij}$$

4. **Fraction Correct (FC):** Fraction of goals having the predicted label same as the gold-standard label, including `NONE`.

$$FC = \frac{\sum_{i=0}^{K} C_{ii}}{\sum_{i=0}^{K} \sum_{j=0}^{K} C_{ij}}$$

Using above measures, we calculate micro-averaged precision, recall and F-measure.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Similar set of measures ($TP, FP, FN, FC, Precision, Recall$ and $F-measure$) are defined considering number of assignments instead of distinct goals.

5.5 Baseline Experiments

We compare our algorithms with basic similarity measures with given thresholds. We compare it with classifiers based on standard similarities like Cosine, Dice and Jaccard. Modified "Developer" template with 21 distinct class labels is fed as the training data for the different baseline experiments.
Cosine Similarity: Each goal description is represented with a TF-IDF based word vector. Similarity between any two goal descriptions g_1 and g_2 is computed as follows:

$$CosineSimilarity(g_1, g_2) = \frac{\vec{wv}_1 \cdot \vec{wv}_2}{\|\vec{wv}_1\| \|\vec{wv}_2\|}$$

where $\vec{wv}_1$ and $\vec{wv}_2$ are word vector representations of g_1 and g_2, respectively.
Dice & Jaccard Similarity: Each goal description is initially cleaned as explained earlier in the section 5.1 and represented with a set of words contained in the clean description.

$$DiceSimilarity(g_1, g_2) = \frac{2 \cdot |S_1 \cap S_2|}{|S_1| + |S_2|}$$

$$JaccardSimilarity(g_1, g_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

where S_1 and S_2 are set of words representing g_1 and g_2, respectively.

For the similarity based classifiers, we compare the every manually created goal with each template goal and assign the corresponding class label of the most similar template goal. We define thresholds δ_{cosine}, δ_{dice} and $\delta_{jaccard}$ for assigning class labels to only highly similar goals. All the manually created having the maximum similarity lower than the threshold, are assigned the `NONE` class. This increases the precision of the classifier when validated with the Gold Standard dataset.

For completeness of our baseline implementations, we have also used two off the shelf classifiers, Nave Bayes and Maximum Entropy classifiers. Since the Nave Bayes builds on word probabilities and occurrences, we consider all the words when calculating the TF-IDF for the words and thereafter the word vectors. This pre-processing of data is provided so the model for the classes during training phase is built based on the entire corpus rather than the considering the limited set of words in the modified "Developer" template. We use the implementation present in Weka [**?**] to train and predict the classes for the manually created goals.

5.6 Results

We compare our approaches with various baselines by computing the evaluation measures defined in the section 5.4. It can be seen in the Table 2 shows that our approaches have clearly outperformed the baseline methods. The best F-measure is reported by the method 2 whereas method 1 reports the best recall among all the methods. Our methods report significantly better results when number of assignments are considered as shown in the Table 3.

Approach	**Precision**	**Recall**	**F1**	**FC**
Dice	0.55	0.53	0.54	0.45
Jaccard	0.56	0.53	0.54	0.45
Cosine	0.59	0.52	0.56	0.47
Nave Bayes	0.58	0.50	0.54	0.42
Max-Ent	0.54	0.56	0.55	0.44
Method 1	0.73	**0.67**	0.70	0.61
Method 2	**0.86**	0.60	**0.71**	**0.70**

Table 2: Comparative Performance of all approaches considering distinct goals in the Gold-standard dataset

Approach	**Precision**	**Recall**	**F1**	**FC**
Dice	0.71	0.68	0.69	0.58
Jaccard	0.71	0.68	0.69	0.58
Cosine	0.70	0.65	0.66	0.56
Nave Bayes	0.58	0.52	0.55	0.45
Max-Ent	0.53	0.49	0.51	0.40
Method 1	0.91	**0.95**	**0.93**	**0.88**
Method 2	**0.94**	0.78	0.85	0.81

Table 3: Comparative Performance of all approaches considering number of goal assignments in the Gold-standard dataset

5.7 Discussion

Method 1 based on Co-training framework achieves more recall and coverage. Unlike method 2, it requires no supervision and makes use of additional information from self comments. Consider the goal description: `Effort in process improvement initiatives`. Method 1 is able to match it correctly to the template goal `Re-engineering saves` even

Template Goal	**Matched Manually Created Goal**
`# of Process / Technical / Domain Related Competencies required for the Role`	`On the track learning of relevant technologies(as applicable like:Flex,Drool,Java.ETL,Tibco,SQL)`
`Contribution to Focus Groups`	`Time spent in implementing account level initiatives per quarter`
`Number of Consulting Engagements/ New Project Wins thru Innovative Ideas`	`No. of unsolicited proposals leading to the revenue growth`
`% SLA compliance/ Remedy Compliance`	`Root Causes in TTs as per Incident Mgmt SOP. Drive for self and onsite and offshore team.`
`Number of Consulting Engagements/ New Project Wins thru Innovative Ideas`	`Number of Demand or Bid Management or Pre- Sales Support (Demos, RFPs) participated and contributed effectively`
`Succession/ Fluidity Planning`	`Number of working Backup groomed & cross transitiion achieved with resource optimization`

Table 4: Examples of manually created Goals matched with their most suitable Template Goals

though there are no explicit keywords for this template goal in the goal description. Self comments provide the knowledge that `process improvement` is semantically similar to `Re-engineering saves`. Likewise, more keywords are from self comments which results in better recall and coverage.

Method 2 achieves better precision and F-measure (distinct) as compared to method 1 as it uses semantic similarity and weak supervision based on active learning. Consider the goal description: `Contribution of articles, discussions to Knowledge sharing platform`. Method 2 matches it correctly to the template goal `No. of reusable components developed/ deployed` whereas method 1 makes an incorrect match to the template goal `Contribution to Focus Groups`. This is because self comments may introduce some noise which may mislead method 1 in making a wrong prediction.

Table 4 shows some examples of manually created goals for various roles in the organization that are matched to relevant template goals. It can be observed that in spite of not having exact word match with the template goals, our methods were successful in matching manually created goals with appropriate template goals. We have got the results validated from HR domain experts.

5.8 Tuning of Parameters for Method 1 (Co-Training Approach)

Since the co-training based approach has a large number of parameters, it is important to empirically determine the best set of parameter values. For this, we randomly divide the gold standard dataset into 5 parts of 200 goals each. Any one of these parts is treated as a "validation" dataset and a set of parameters leading to the best performance in terms of F-measure (distinct) is chosen. Then the classification performance using the same of parameters is measured on the remaining 800 goals. This is repeated for each part of 200 goals identified as "validation" and all the performance measures reported are averaged.

We have tuned the following parameters (in the Algorithm 2) using the above mentioned tuning process.

1. I : The co-training process stops if no new additions in labelled data happens in any iteration. To limit the training time, a limit I on number of iterations is used.

2. J : In the initial iterations of co-training, it is undesirable to add incorrect predictions to the labelled data. In the initial J iterations where $J < I$, we update the labelled data with the classified data point only if both the classifiers predict the same class label with confidences greater than the respective thresholds, η_1, η_2.

3. η_1, η_2 : These are the confidence thresholds for classifiers C_1 and C_2 respectively to add goals into the labelled data during co-training.

4. ADD_{max}, ADD_{min} : These are used to address the problem of Class Imbalance as described in the section 3.4.

5. η_1', η_2' : During co-training, if the condition of ADD_{min} is not satisfied, we reduce the original confidence thresholds to η_1' and η_2'.

6. $\eta^a, \eta_1^f, \eta_2^f$: After co-training, the final predictions for the unlabelled goals are determined using these thresholds.

The best combination of parameters after tuning is : $I = 5, J = 3, \eta_1 = 0.8, \eta_2 = 0.8, ADD_{max} = 10, ADD_{min} = 1, \eta_1' = 0.3, \eta_2' = 0.4, \eta^a = 0.2, \eta_1^f = 0.4, \eta_2^f = 0.5$

The number of iterations are chosen arbitrarily. We have done empirical analysis to identify a correct threshold for the Maximum number of iterations required. We have noticed that the increasing the maximum number of iterations for training has actually classified more goals. But this reduces the F1-Measure of Distinct Goals significantly. The more number of iterations aren't improving the accuracy. but lower iterations will result in poor coverage. Figure 3 shows change in F-measure (distinct) by varying I and setting values of all other parameters to the identified best combination. Empirical results show that $I = 5$ provides a balance between the trade off parameters.

The limits on the number of goals per class (ADD_{min} and ADD_{max}) that can be added reduces the class imbalance that might be introduced in the training iterations. We have performed experiments by increasing the maximum number of goals that can be added to a class in each iteration. This will increase the fraction of goals that are bound to be classified, but this increase is at the cost of reduction in precision and recall. Figure 4 shows change in F-measure (distinct) by varying ADD_{max} and setting values of all other parameters

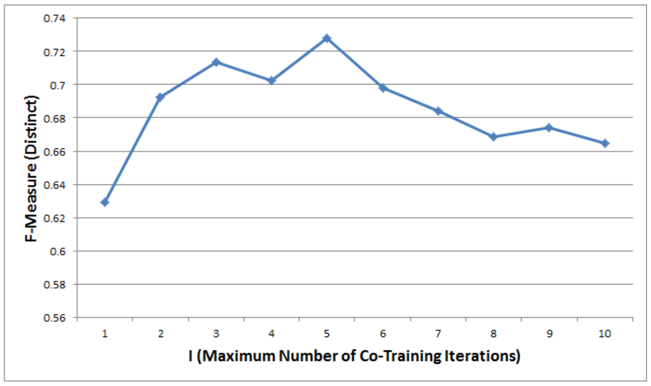

Figure 3: Change in F-measure (distinct) for various values of I (Maximum number of co-training iterations)

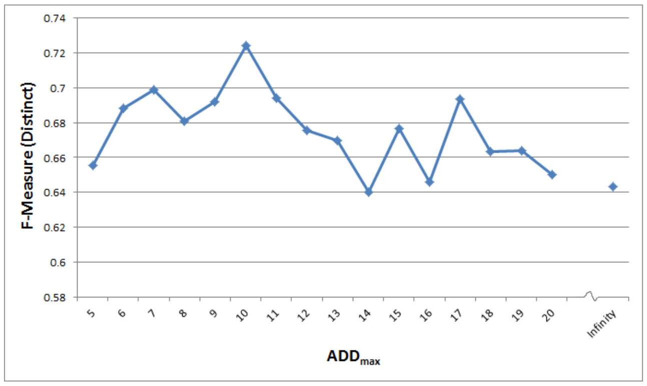

Figure 4: Change in F-measure (distinct) for various values of ADD_{max}

to the identified best combination. It can be seen that the F-measure (distinct) achieves the maximum at $ADD_{max} = 10$ and tends to decrease thereafter.

6. CONCLUSION AND FUTURE WORK

We highlighted the need for standardization of goals in the Performance Appraisal process. We posed this problem as a multi-class classification problem and explored two different approaches with minimal human supervision. The first approach, based on Co-training framework achieves better recall and coverage. This is due to the use of additional knowledge acquired from the self comments. The second approach uses semantic similarity and weak supervision using active learning. It achieves the best precision and F-measure (distinct) as compared to other methods. Both the approaches clearly outperform multiple baseline methods.

From domain perspective, it is important to classify maximum number of manually created goals resulting higher level of standardization of goals. Hence, we also calculated the performance measures based on number of assignments of each goal.

In future, we plan to extend this work to propose new goals to be added to the template by grouping "semantically" similar goals which are not matched to any of the existing template goals.

7. REFERENCES

[1] Apache opennlp. *https://opennlp.apache.org/*.

[2] Jazzy : The java open source spell checker. *http://jazzy.sourceforge.net/*.

[3] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[4] Victoria Bobicev and Marina Sokolova. An effective and robust method for short text classification. In *AAAI*, pages 1444–1445, 2008.

[5] B.G. Buchanan and E.H. Shortliffe. *Rule Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984. ISBN 978-0-201-10172-0.

[6] Mengen Chen, Xiaoming Jin, and Dou Shen. Short text classification improved by learning multi-granularity topics. In *IJCAI*, pages 1776–1781. Citeseer, 2011.

[7] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM, 2008.

[8] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998 (ed.).

[9] Gabriel Pui Cheong Fung, Jeffrey X Yu, Hongjun Lu, and Philip S Yu. Text classification without negative examples revisit. *Knowledge and Data Engineering, IEEE Transactions on*, 18(1):6–20, 2006.

[10] Rayid Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *ICML*, volume 2, pages 8–12, 2002.

[11] D. Giorgetti and F. Sebastiani. Automating survey coding by multiclass text categorization techniques. *Journal of the American Society for Information Science and Technology*, 54(12):1269–1277, 2003.

[12] Daniel Jurafsky and James Martin. *Speech and Natural Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Education Inc., 2009. ISBN 9780131873216.

[13] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: An introduction to cluster analysis*. Wiley series in Probability and Statistics. John Wiley and Sons, New York, 1990.

[14] Hang Li and Kenji Yamanishi. Mining from open answers in questionnaire data. In *Proceedings of Seventh ACM SIGKDD*, 2001.

[15] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592, 2003.

[16] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Building text classifiers using positive and unlabeled examples. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 179–186. IEEE, 2003.

[17] Zhengdong Lu and Hang Li. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*, pages 1367–1375, 2013.

[18] Roberto Navigli. Word sense disambiguation: A

survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

[19] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM, 2000.

[20] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2-3):103–134, 2000.

[21] S. Patil and G. K. Palshikar. Surveycoder: A system for classification of survey responses. In *Proceedings of the 18th International Conference on Application of Natural Language to Information Systems (NLDB 2013), LNCS 7934*. Springer-Verlag, 2013.

[22] S. Patil and B. Ravindran. Active learning based weak supervision for textual survey response classification. In *In Proceedings of 16th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing), Part II, LNCS 9042*. Springer, 2015.

[23] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[24] B. Settles. *Active Learning*. Morgan Claypool, Synthesis Lectures on AI and ML, 2012.

[25] Bharath Sriram. *Short text classification in Twitter to improve information filtering*. PhD thesis, The Ohio State University, 2010.

[26] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842. ACM, 2010.

[27] Aixin Sun. Short text classification using very few words. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1145–1146. ACM, 2012.

[28] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, Upper Saddle River, NJ, 2005.

[29] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[30] Sarah Zelikovitz and Haym Hirsh. Improving short text classification using unlabeled background knowledge to assess document similarity. In *Proceedings of the seventeenth international conference on machine learning*, volume 2000, pages 1183–1190, 2000.

[31] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.