Grammatical Error Correction by Crowdsourcing Post-Editors

Ziyi Yang, Kun Peng, and Rui Yan
School of Engineering and Applied Science,
University of Pennsylvania, Philadelphia, PA 19104, USA

1 Introduction

Grammatical error correction has arisen as an important component for crowdsourcing translation tasks, especially when we need to hire non-native bilingual workers. The large and growing number of speakers of English as a second language has motivated the research direction, and researchers have been investigated possible solutions to help these non-native students and professionals improve the grammaticality, fluency, and overall quality of their English writing. The recent CoNLL shared task on grammatical error correction has produced an even greater wave of interest in developing systems to correct non-native speaker errors. In machine translation, researchers have experimented with automatic post-editing to improve system output and hence aid the work of human translators.

The error correction problem could be addressed by non-experts, but preferably by native speakers, since the task requires only fluency in English. Crowdsourcing has been used successfully for proofreading tasks, but quality control in these applications is ultimately performed manually by the user, making it harder to scale. In this study, we propose some methods to do quality control automatically: we hope to select the best qualified correction operations for a particular source sentence to revise.

2 Methods

We establish tuples from the original data files, the format is as follows:

$$<workerID, transID, translation, operations>$$

This tuple means that the *workerID* have made *operations* of corrections to the source sentence (*translation*) with the *transID*. With the elements extracted from the tuples from the original data, we could build operation graphs for the feasible corrections to take.

2.1 Random

We randomly select the operations from the workers without excessive calibrating.

2.2 Operation Graph Establishment

In this method, we apply each separate operation onto a source sentence, i.e., a translation, to get a modified candidate sentence. Then we frame the candidate sentences into a graph, modeling the relationships between sentences, i.e., semantic similarity. Let G denote the graph with nodes V and edges E, and G = (V,E), which is a weighted undirected graph representing the candidate translations and their correlations (represented by the textual similarity between the translated sentences). Note that we only establish linkage between the candidate translation groups for the same source sentences. In other words, we do not establish textual linkage between candidate translations for different source sentences to translate. The reason is that different groups of source sentences should not have influence on each other.

In this sense, we have a series of isolated subgraphs, each represents a source sentence with a certain atomic operation with linkage established inside the sub-graph. We run random walks on these subgraphs. For every sub-graph, a random walk on a graph is a Markov chain, its states being the vertices of the graph. It can be described by a square matrix, where the dimension is the number of vertices in the graph. The stochastic matrix prescribing the transition probabilities from one vertex to the next. Within each sub-graph, the score of candidate translations are calculated as follows:

$$\mathbf{c} = (1-\mu)M^T\mathbf{c} + \mu\frac{1}{|V_c|} \quad (1)$$

2.3 Co-ranking Framework based on Worker Agreement

Finally, we include the post-editors and their collaboration into the graph framework. We propose our method operates over a heterogeneous

network that includes workers and corrected sentences. We frame both components into graphs, using relationships to connect these parts as a ranking paradigm. Let G denote the heterogeneous graph with nodes V and edges E, and G = (V,E) = ($V_C, V_T, E_C, E_T, E_{CT}$). G is divided into three subgraphs, G_T, G_C, and G_{CT}. G_C = (V_C, E_C) is a weighted undirected graph representing the candidate translations and their relationships. Let V_C = $\{c_i|c_i \in V_C\}$ denote a collection of $|V_C|$ translated and edited sentences, and E_C the set of linkage representing affinity between them, established by textual similarity between the translated sentences. G_T = (V_T, E_T) is a weighted undirected graph representing the editing agreement among Turkers. V_T = $\{t_i|t_i \in V_T\}$ is the set of working pairs with size $|V_T|$. Links E_T among Turkers are established by their operation agreements, namely how many operations in common for two workers would commit. Each operation from each worker would produce an output translation. $G_{CT} = (V_{CT}, E_{CT})$ is an unweighted bipartite graph that ties G_T and G_C together and represents "authorship". The graph G consists of nodes V_{CT} = $V_T \cup V_C$ and edges E_{CT} connecting each candidate with its generators. Typically, a candidate is generated by a post-editor, but note that it is still possible to share the common authorship when two workers commit the same operation(s).

• We use adjacency matrix $[M]_{|\mathbf{c}|\times|\mathbf{c}|}$ to describe the homogeneous affinity between candidates and $[N]_{|\mathbf{t}|\times|\mathbf{t}|}$ to describe the affinity between Turkers.

$$\mathbf{c} \propto M^T\mathbf{c}, \qquad \mathbf{t} \propto N^T\mathbf{t} \tag{2}$$

where $c = |V_C|$ is the number of vertices in the candidate graph and $t = |V_T|$ is the number of vertices in the Turker graph. The adjacency matrix $[M]$ denotes the transition probabilities between candidates, and analogously matrix $[N]$ denotes the affinity between Turker agreements.

• We use an adjacency matrix $[\hat{W}]_{|\mathbf{c}|\times|\mathbf{t}|}$ and $[\bar{W}]_{|\mathbf{t}|\times|\mathbf{c}|}$ to describe the authorship between the output candidate and the producer Turker pair from both of the candidate-to-Turker and Turker-to-candidate perspectives.

$$\mathbf{c} \propto \hat{W}^T\mathbf{t}, \qquad \mathbf{t} \propto \bar{W}^T\mathbf{c} \tag{3}$$

Step 1: compute the saliency scores of candidates, and then normalize using ℓ-1 norm.

$$\begin{aligned}\mathbf{c}^{(n)} &= (1-\lambda)M^T\mathbf{c}^{(n-1)} + \lambda\hat{W}\mathbf{t}^{(n-1)} \\ \mathbf{c}^{(n)} &= \mathbf{c}^{(n)}/||\mathbf{c}^{(n)}||_1\end{aligned} \tag{4}$$

Step 2: compute the saliency scores of Turker pairs, and then normalize using ℓ-1 norm.

$$\begin{aligned}\mathbf{t}^{(n)} &= (1-\lambda)N^T\mathbf{t}^{(n-1)} + \lambda\bar{W}\mathbf{c}^{(n-1)} \\ \mathbf{t}^{(n)} &= \mathbf{t}^{(n)}/||\mathbf{t}^{(n)}||_1\end{aligned} \tag{5}$$

where λ specifies the relative contributions to the saliency score trade-off between the homogeneous affinity and the heterogeneous affinity. In order to guarantee the convergence of the iterative form, we must force the transition matrix to be stochastic and irreducible. To this end, we must make the **c** and **t** *column stochastic*. **c** and **t** are therefore normalized after each iteration of equations.

2.3.1 Intra-Graph Ranking

The standard PageRank algorithm starts from an arbitrary node and randomly selects to either follow a random out-going edge (considering the weighted transition matrix) or to jump to a random node (treating all nodes with equal probability).

In a simple random walk, it is assumed that all nodes in the transitional matrix are equi-probable before the walk starts. Then **c** and **t** are calculated as:

$$\mathbf{c} = \mu M^T\mathbf{c} + (1-\mu)\frac{\mathbf{1}}{|V_C|} \tag{6}$$

and

$$\mathbf{t} = \mu N^T\mathbf{t} + (1-\mu)\frac{\mathbf{1}}{|V_T|} \tag{7}$$

where **1** is a vector with all elements equaling to 1 and the size is correspondent to the size of V_C or V_T. μ is the damping factor usually set to 0.85, as in the PageRank algorithm.

2.3.2 Affinity Matrix Establishment

We introduce the affinity matrix calculation, including homogeneous affinity (i.e., M, N) and heterogeneous affinity (i.e., $\hat{W}, \bar{W}$).

As discussed, we model the collection of candidates as a weighted undirected graph, G_C, in which nodes in the graph represent candidate sentences and edges represent lexical relatedness. We define an edge's weight to be the cosine similarity between the candidates represented by the nodes that it connects. The adjacency matrix M describes such a graph, with each entry corresponding to the weight of an edge.

$$\begin{aligned}\mathcal{F}(c_i, c_j) &= \frac{c_i \cdot c_j}{||c_i||||c_j||} \\ M_{ij} &= \frac{\mathcal{F}(c_i, c_j)}{\sum_k \mathcal{F}(c_i, c_k)}\end{aligned} \tag{8}$$

where $\mathcal{F}(.)$ is the cosine similarity and c is a term vector corresponding to a candidate. We treat a candidate as a short document and weight each term with *tf.idf*, where *tf* is the term frequency and *idf* is the inverse document frequency.

The worker graph, G_T, is an undirected graph whose edges represent "agreement." Formally, let t_i and t_j be two editors, there is an edge between t_i and t_j) if t_i and t_j share some agreed operations to a particular sentence. Let the function $\mathcal{I}(t_i, t_j)$ denote the number of "agreements" ($\#agreement$) between t_i and t_j.

$$\mathcal{I}(t_i, t_j) = \begin{cases} \#agreement & (e_{ij} \in E_T) \\ 0 & \text{otherwise} \end{cases}, \quad (9)$$

Then the adjacency matrix N is then defined as

$$N_{ij} = \frac{\mathcal{I}(t_i, t_j)}{\sum_k \mathcal{I}(t_i, t_k)} \quad (10)$$

In the bipartite candidate-Turker graph G_{TC}, the entry $E_{TC}(i, j)$ is an indicator function denoting whether the candidate c_i is generated by t_j:

$$\mathcal{A}(c_i, t_j) = \begin{cases} 1 & (e_{ij} \in E_{TC}) \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

Through E_{TC} we define the weight matrices $\bar{W}_{ij}$ and $\hat{W}_{ij}$, containing the conditional probabilities of transitions from c_i to t_j and vice versa:

$$\begin{aligned} \bar{W}_{ij} &= \frac{\mathcal{A}(c_i, t_j)}{\sum_k \mathcal{A}(c_i, t_k)}, \\ \hat{W}_{ij} &= \frac{\mathcal{A}(c_i, t_j)}{\sum_k \mathcal{A}(c_k, t_j)} \end{aligned} \quad (12)$$

After the co-ranking process, we could get the ranking score for each candidate sentence (which means a single operation), and we select the operations with the higher ranks than the others.

3 Evaluation Metric and Performance

3.1 Precision, Recall and F

As the ground truth data provides the gold operations from the CoNLL shared task, we could actually apply the standard evaluation of *Precision*, *Recall* and *F-measure*. For precision, we need to check how many selected operations are actually in the gold answers, and for recall, we need to examine that how many gold operations are finally retrieved by our methods. F-measure is the harmonious average of both precision and recall.

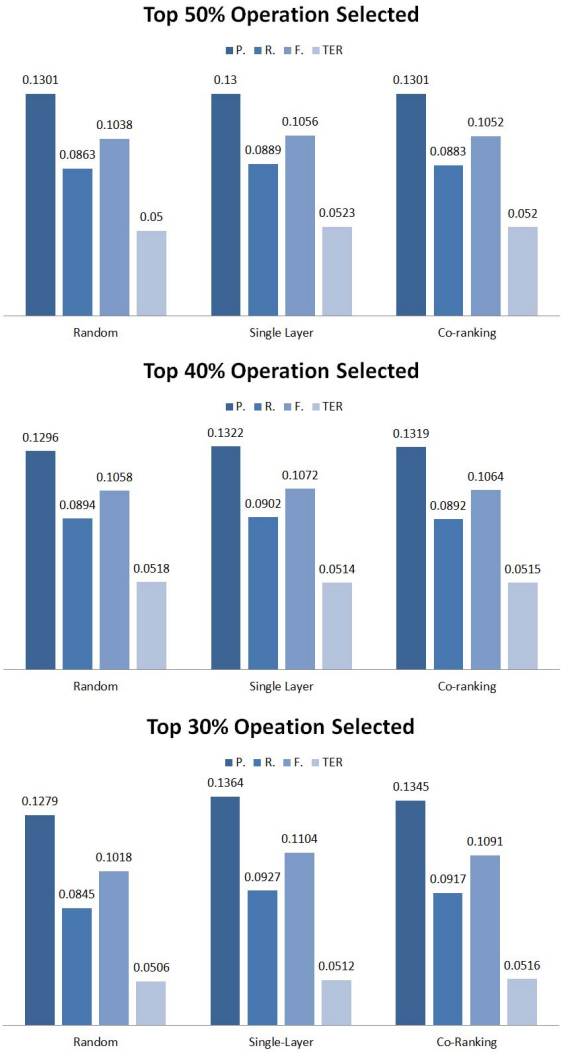

Figure 1: Precision, recall, F-score and TER performance for 30%, 40% and 50% operations.

3.2 TER Edit Distance

The precision, recall and F-measure based evaluation metric could possibly be too rigid for grammatical corrections, since for humans there are generally multiple ways to make revision rather than a standard single answer, e.g., maybe different terms with similar meanings could also make a good correction. Hence we include the TER edit distance as an alternative evaluation metric. Intuitively, the closer edit distance for a candidate sentence after applying a certain operation compared to the ground truth sentences, the better candidate it would be.

3.3 Performance

We rank all possible operations, and select the corresponding top-k operations, w.r.t $k = 30\%, 40\%, 50\%$. Results are shown in Figure 1.