Supplementary material for

Rohan Ramanath, Monojit Choudhury, Kalika Bali and Rishiraj Saha Roy, "Crowd Prefers the Middle Path: A New IAA Metric for Crowdsourcing Reveals Turker Biases in Query Segmentation", in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL '13), Sofia, Bulgaria, 4 – 9 August 2013.

and

Rohan Ramanath, Monojit Choudhury and Kalika Bali, "Entailment: An Effective Metric for Comparing and Evaluating Hierarchical and Non-hierarchical Annotation Schemes", in Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse (LAW VII), Sofia, Bulgaria, 8 – 9 August 2013.

1 Introduction

This article includes supplementary material for the papers listed above. Section 2 describes the released dataset. Worked out examples and pseudocodes for computing Krippendorff's Alpha are presented in Section 3. Section 4 explains entailment in detail. Section 5 mentions the relevant license agreement. Please address all queries to any of the authors of the above papers.

2 Datasets

Segmentation annotations of the **Q500**, **QG500**, **Q700**, **S300** and **QRand** datasets [1] are contained in the accompanying folder named "Datasets". All the data files are released in JSON[1] format (similar to XML) in order to allow easy interoperability between the data and code. The naming convention used is ⟨*dataset_name*⟩*_flat.json* for flat segmentation and ⟨*dataset_name*⟩*_nested.json* for nested segmentation. Each of the provided JSON files has got two keys – an *item* and an *annotation set*. The *items* indicate the number of queries or sentences in the given dataset. The *annotation set* is a JSON object where the key is a query or sentence and the value is a JSON object consisting of up to ten annotations for the given key. Dataset details are summarized in Table 1.

3 Computation of Krippendorff's Alpha

In this section, we will work out the computation of Krippendorff's α for the toy dataset of two queries and their annotations as shown in Table 2. The first query has three annotations and the second has two annotations. The annotations refer to the boundary

[1] `http://www.json.org/`

File name	**Content**
Q500_flat.json	Flat segmentations for **Q500** query set [1] by Turkers
Q500_nested.json	Nested segmentations for **Q500** query set [1] by Turkers
QG500_flat.json	Flat segmentations for **Q500** [1] by gold annotators
QG500_nested.json	Nested segmentations for **Q500** [1] by gold annotators
Q700_flat.json	Flat segmentations for **Q700** query set [1] by Turkers
Q700_nested.json	Nested segmentations for **Q700** query set [1] by Turkers
S300_flat.json	Flat segmentations for **S300** sentence set [1] by Turkers
S300_nested.json	Nested segmentations for **S300** sentence set [1] by Turkers
U250_flat.json	Flat segmentations of unigram model generated queries by Turkers
U250_nested.json	Nested segmentations of unigram model generated queries by Turkers
B250_flat.json	Flat segmentations of bigram model generated queries by Turkers
B250_nested.json	Nest segmentations of bigram model generated queries by Turkers
T250_flat.json	Flat segmentations of trigram model generated queries by Turkers
T250_nested.json	Nested segmentations of trigram model generated queries by Turkers

Table 1: Dataset summary.

Query	**Annotation 1**	**Annotation 2**	**Annotation 3**
`major league baseball salary cap`	1 0 2 0	0 1 0 2	2 0 1 0
`wind beneath my wings sheet music`	0 1 0 2 0	0 2 0 1 0	-

Table 2: Toy dataset of two queries used for the computing Krippendorff's α.

values of nested segmentation computed as described by Ramanath et. al. [1]. Computation of α consists of the following three steps: (1) Computation of the variance within the annotations of a selected query, (2) Computation of the variance between all annotations in the dataset, and (3) Computation of α.

3.1 Variance between annotations for one query

Let us take the first query Q_1, `major league baseball salary cap`. Here, the number of words is five and thus the number of possible flat segments is four. For every pair of annotations (A_i, A_j) represented as boundary variables, we find the distance $d_2(A_i, A_j)$ as described in Section 4.2 of Ramanath et. al. [1]. The distance d_1 can be computed similarly (steps to compute d_1 and d_2 are provided in Algorithms 1 and 2). In our example, $A_1 = [1, 0, 2, 0]$, $A_2 = [0, 1, 0, 2]$ and $A_3 = [2, 0, 1, 0]$. The d_2 values between the following annotation pairs are given below.

$$d_2(A_1, A_2) = (|1-0| + |0-1| + |4-0| + |0-4|)/4 = 2.5$$
$$d_2(A_1, A_3) = (|1-4| + |0-0| + |4-1| + |0-0|)/4 = 1.5$$
$$d_2(A_2, A_3) = (|0-4| + |1-0| + |0-1| + |4-0|)/4 = 2.5$$

Using these distance values, we can fill the *within matrix* M_1 for Q_1 as shown below. A cell $\langle i, j\rangle$ of this matrix represents the distance between annotations represented by row i and column j.

Algorithm 1 Distance Metric d_1

```
1: $l \leftarrow len(A)$
2: $sum \leftarrow 0$
3: for $i = 0$ to $l - 1$ do
4:     $sum \leftarrow sum + abs(A[i] - B[i])$
5: end for
6: return $sum$
```

Algorithm 2 Distance Metric d_2

```
1: $l \leftarrow len(A)$
2: $sum \leftarrow 0$
3: for $i = 0$ to $l - 1$ do
4:     $sum \leftarrow sum + abs(A[i]^2 - B[i]^2)$
5: end for
6: return $sum$
```

	A_1	A_2	A_3
A_1	0.00	2.50	1.50
A_2	2.50	0.00	2.50
A_3	1.50	2.50	0.00

Here c, the number of annotations for a query, is 3. The sum of all values in matrix M_1, $sum(M_1)$ is 13. We define a quantity $within\_sum\_query$, computed for every query, which is defined as $sum(M_1)/(c * (c-1))$. So, for Q_1, $within\_sum\_Q_1 = 13/(3*2) = 2.17$.

For the second query, `wind beneath my wings sheet music` (Q_2), the number of words is six and the number of possible segments is five. The annotations are $A_1 = [0, 1, 0, 2, 0]$ and $A_2 = [0, 2, 0, 1, 0]$.

$$d_2(A_1, A_2) = (|0| + |-3| + |0| + |3| + |0|)/5 = 1.2$$

Using these distance values, we can fill the $within\_matrix$ M_2 for Q_2.

	A_1	A_2
A_1	0.00	1.20
A_2	1.20	0.00

Here the value of c is two. The sum of the $within\_matrix$ is 2.4. Thus, here, $within\_sum\_Q_2 = 1.2$. We now need to add up the $within\_sum\_query$ for every query, giving $within\_sum\_total$, which, here is equal to $2.17 + 1.2 = 3.37$. The variance within annotations of a query, over the entire dataset, is defined as

$$Variance(within) = within\_sum\_total/(2*q)$$

where q is the total number of queries. The value of this variance in this context is thus $3.37/(2*2) = 0.84$.

Algorithm 3 $withinVariance(NDict)$

```
1: global $withinDict$    ▷ Dictionary to store variance within annotations of a given query
2: $sum \leftarrow 0$
3: for $query\ q$ in $NDict$ do
4:     $segments \leftarrow Ndict[q]$
5:     $l \leftarrow words(q)$    ▷ Number of words in $q$
6:     $n \leftarrow len(segments)$
7:     for $i = 0$ to $n$ do
8:         for $j = 0$ to $i + 1$ do
9:             $A \leftarrow segments[i]$
10:            $B \leftarrow segments[j]$
11:            $m[i, j] \leftarrow d_2(A, B)/(l - 1)$
12:            $m[j, i] \leftarrow m[i, j]$
13:        end for
14:    end for
15:    $withinDict[q] \leftarrow$ m
16:    $c \leftarrow rows(m)$    ▷ Number of rows in $m$
17:    $m_{sum} \leftarrow sum(m)$    ▷ Sum all cells of $m$
18:    $sum \leftarrow sum + m_{sum}(c * (c - 1))$
19: end for
20: $q_{len} \leftarrow len(withinDict)$
21: return $sum/(2 * q_{len})$
```

Steps for computing $Variance(within)$ are formalized in Algorithm 3. $NDict$ is a dictionary containing all the queries in the dataset as keys and a list of boundary values of the corresponding nested segmentation as values.

3.2 Variance over all annotations in a dataset

Similarly, the distance matrix is computed for all pairs of annotations across queries. When the two queries are of the same length, the distance between their annotations can be calculated in the same way as two annotations for the same query. The difference occurs when queries are of different lengths, in which case we find the distance between them as shown below. Let us consider A_1 from Q_1 ($A_{11} = [1, 0, 2, 0]$) and A_1 from Q_2 ($A_{21} = [0, 1, 0, 2, 0]$). The X_i-s, as shown below, represent distinct configurations of the annotations under consideration.

$$\begin{array}{lcccccccl} A_{21} & = & 0 & 1 & 0 & 2 & 0 & (X_1) \\ A_{11} & = & 1 & 0 & 2 & 0 & & (X_2) \\ A_{11} & = & & 1 & 0 & 2 & 0 & (X_3) \end{array}$$

$$d_2(X_1, X_2) = (|0 - 1| + |1 - 0| + |0 - 4| + |4 - 0|)/4 = 2.5$$
$$d_2(X_1, X_3) = (|0 - 0| + |0 - 0| + |0 - 0| + |0 - 0|)/4 = 0$$
$$d_2(A_{11}, A_{21}) = (2.5 + 0)/2 = 1.25$$

Similarly, the remaining entries of the distance matrix between the annotations of Q_1 and Q_2, say, M_3, can be calculated. The completed matrix M_3 is given below.

$$\begin{array}{cccc} & A_{11} & A_{12} & A_{13} \\ A_{21} & 1.25 & 1.25 & 2.00 \\ A_{22} & 2.00 & 2.00 & 1.25 \end{array}$$

The sum of the values in M_3 is 9.75. We now construct a new matrix representing the overall distance between Q_1 and Q_2, say, M_4. The entries for the diagonal elements are the corresponding sum of values of the $within\_matrices$.

$$\begin{array}{ccc} & Q_1 & Q_2 \\ Q_1 & 13.0 & 9.75 \\ Q_2 & 9.75 & 2.40 \end{array}$$

The sum of the values in M_4 is termed as $between\_sum$ for Q_1 and Q_2, which is 34.9 here. Let the total number of annotations in the dataset be qc. Here, $qc = 3 + 2 = 5$. We define $Variance(between)$ as the variance between all annotations of all queries in the dataset as below:

$$Variance(between) = between\_sum/(2 * qc * (qc - 1))$$

The value of $Variance(between)$ for our example is thus 0.87. The algorithm for computing $Variance(between)$ is presented in two parts (Algorithms 4 and 5).

3.3 Computation of alpha

Finally, Krippendorff's α is defined as

$$\alpha = 1 - [Variance(within)/Variance(between)]$$

Using the above definition, we can compute its value to be $1-(0.84/0.87) = 0.035$. The corresponding pseudocode is provided in Algorithm 6.

4 Entailment

In this section, we describe an equivalent definition of *entailment* [2], presented by Algorithm 7, which can be used to automatically detect whether a given nested segmentation *entails* a flat segmentation. Let *i* be the position corresponding to the highest node of the nested segmentation tree. If position *i* in the corresponding flat segmentation does not have a boundary, but there is a boundary to the right or left of *i*, then algorithm textitisEntail returns *False*, signifying that the flat and nested segmentations do not entail each other. On the contrary, if position *i* in the flat segmentation does indicate a boundary, we recurse by calling *isEntail* on the left and right subtrees of the original nested segmentation tree (with the corresponding *spliced* flat segmentations). All segmentations of unit length entail each other, and hence, *isEntail* returns *True* for segmentations of unit length. This, in turn, serves as the base case for the above recursion.

Algorithm 4 $between(Q_1, Q_2, NDict)$

```
global $betweenDict$ ▷ Dictionary to store variance between the annotations of all pairs of queries
$l_1 \leftarrow words(Q_1)$                                  ▷ Number of words in $Q_1$
$l_2 \leftarrow words(Q_2)$                                  ▷ Number of words in $Q_2$
if $l_1 = l_2$ then
    $s_1 \leftarrow Ndict[Q_1]$
    $s_2 \leftarrow Ndict[Q_2]$
    $n_{s_1} \leftarrow len(s_1)$
    $n_{s_2} \leftarrow len(s_2)$
    for $i = 0$ to $n_{s_1}$ do
        for $j = 0$ to $n_{s_2}$ do
            $m[i, j] \leftarrow d_2(s_1[i], s_2[j])/(l_1 - 1)$
        end for
    end for
else
    $l_{min} \leftarrow min(l_1, l_2)$
    if $l_{min} = l_1$ then
        $fixed \leftarrow Q_1$
        $move \leftarrow Q_2$
    else
        $fixed \leftarrow Q_2$
        $move \leftarrow Q_1$
    end if
    $s_f \leftarrow NDict[fixed]$
    $s_m \leftarrow NDict[move]$
    $iter \leftarrow abs(l_1 - l_2)$
    $n_{s_f} \leftarrow len(s_f)$
    $n_{s_m} \leftarrow len(s_m)$
    for $i = 0$ to $n_{s_f}$ do
        for $j = 0$ to $n_{s_m}$ do
            $cursum \leftarrow 0$
            $count \leftarrow 0$
            for $k = 0$ to $iter + 1$ do
                $s \leftarrow d_2(s_f[i], s_m[j][k \rightarrow k + q])/(l_{min} - 1)$
                $cursum \leftarrow cursum + s$
                $count \leftarrow count + 1$
            end for
            $m[i, j] \leftarrow cursum/count$
        end for
    end for
end if
$betweenDict[Q_1][Q_2] \leftarrow m$
$m_{sum} \leftarrow sum(m)$                                  ▷ Sum all cells of $m$
return $m_{sum}$
```

Algorithm 5 $betweenVariance(NDict)$

```
1:  $sum \leftarrow 0$
2:  $total \leftarrow 0$
3:  **for** $query\ q_1$ **in** $NDict$ **do**
4:      **for** $query\ q_2$ **in** $NDict$ **do**
5:          $pair_{var} \leftarrow between(q_1, q_2, NDict)$
6:          $sum \leftarrow sum + pair_{var}$
7:      **end for**
8:  **end for**
9:  **for** $query\ q_1$ **in** $NDict$ **do**
10:     $total \leftarrow total + len(NDict[q_1])$
11: **end for**
12: **return** $sum/(2 * total * (total - 1))$
```

Algorithm 6 $alpha(NDict)$

```
1: $within_{var} \leftarrow withinVariance(NDict)$
2: $between_{var} \leftarrow betweenVariance(NDict)$
3: $\alpha \leftarrow (1 - (within_{var}/between_{var}))$
4: **return** $\alpha$
```

Algorithm 7 $isEntail(flat, nested)$

```
1:  **if** $len(nested) \leq 1$ or $len(flat) \leq 1$ **then**    ▷ $flat$, $nested$ are lists containing boundary values
2:      **return** True
3:  **end if**
4:  $h \leftarrow largest(nested)$
5:  $i \leftarrow indexOf(h)$
6:  **if** $flat[i] = 1$ **then**
7:      **if not** $isEntail(flat[0 \rightarrow i], nested[0 \rightarrow i])$ **or not** $isEntail(flat[i+1 \rightarrow len(flat)], nested[i+1 \rightarrow len(nested)])$ **then**
8:          **return** False
9:      **else**
10:         **return** True
11:     **end if**
12: **else**
13:     **while** $h \neq 0$ **do**
14:         $nested[i] \leftarrow -nested[i]$
15:         $h \leftarrow largest(nested)$
16:         $i \leftarrow indexOf(h)$
17:         **if** $flat[i] = 1$ **then**
18:             **return** False
19:         **end if**
20:     **end while**
21:     **return** True
22: **end if**
```

5 Dataset License

This section mentions the license associated with the use of the accompanying dataset (Section 2). Use of this freely available dataset implies that the researcher has read and agreed to the terms and conditions of Microsoft Research's End User License Agreement (MSR-EULA). The MSR-EULA is made available along with the dataset and this document. When using the dataset, please cite either (or both) of the two papers listed at the beginning, depending on the relevance to your work.

References

[1] ROHAN RAMANATH, MONOJIT CHOUDHURY, KALIKA BALI AND RISHIRAJ SAHA ROY: *Crowd Prefers the Middle Path: A New IAA Metric for Crowdsourcing Reveals Turker Biases in Query Segmentation*, 51st Annual Meeting of the Association for Computational Linguistics, (2013).

[2] ROHAN RAMANATH, MONOJIT CHOUDHURY, AND KALIKA BALI: *Entailment: An Effective Metric for Comparing and Evaluating Hierarchical and Non-hierarchical Annotation Schemes*, The 7th Linguistic Annotation Workshop and Interoperability with Discourse, (2013).