A Global Voting for Lexicon Learning

In this appendix we provide the step-by-step pseudo-code for the voting strategies described in §4. We first define in §A.1 the function AGGREGATEVOTES to aggregate votes over lexemes (for review of lexemes and factored lexicons, see §2.1). We then describe MAXVOTE in §A.2 and CONSENSUSVOTE in §A.3. Figure 6 provides pseudocode for both strategies. While voting is defined over lexemes, as described in §4, the output of each voting strategy algorithm is a set of lexical entries to add to the lexicon, as expected for the procedure VOTE in line 5 of Algorithm 1.

A.1 Vote Aggregation for Lexemes

Given sets of lexemes, votes are aggregated independently for each of the strings included in the lexemes. Let $L^{(1)}, \ldots, L^{(N)}$ be a collection of N sets of lexemes, and let $L_\cup = \bigcup_{i=1}^{N} L^{(i)}$ be their union. Let $S(.)$ be a function that returns the string(s) for a single lexeme (set of lexemes). Therefore, $S(L_\cup)$ denotes the set of all strings for the N lexeme sets. Given the above definitions, AGGREGATEVOTES$(L^{(1)}, \ldots, L^{(N)})$ computes a voting dictionary $\mathcal{V}$ with an entry $\mathcal{V}_s$ for each string $s \in S(L_\cup)$. Each entry $\mathcal{V}_s$ is a function $L_\cup \to \mathbb{R}$, which given a lexeme $\ell \in L_\cup$ returns:

$$\mathcal{V}_s(\ell) = \sum_{i=1}^{N} \frac{\mathbb{1}\{\ell \in L^{(i)}\}}{\sum_{\ell' \in L_\cup} \mathbb{1}\{\ell' \in L^{(i)} \wedge S(\ell') = s\}}$$

Where $\mathbb{1}\{.\}$ is an indicator function. $\mathcal{V}_s(\ell)$ is the vote assigned to the lexeme ℓ for the string s. Each $L^{(i)}$ may contribute a total vote of 1.0 for each string, which is distributed uniformly among all lexemes containing this string in $L^{(i)}$. See §4 for example votes computation.

A.2 Strategy 1: MAXVOTE

Algorithm 4 provides the pseudocode for the first voting strategy, as described in §4.1. We define the function $\mathcal{L}(.)$ to return the lexeme(s) for a lexical entry (set of lexical entries). We first create the set of all new lexemes L_{new} (line 1), which includes all generated lexemes that don't appear in the model lexicon. Next, the voting dictionary $\mathcal{V}$ is computed for the N lexeme sets using AGGREGATEVOTES (line 2). In lines 4-5, for each string $s \in S(L_{new})$, we find the new lexeme with most votes. In case of a tie, no lexeme is selected. Finally, in lines 6-7, the lexical entries corresponding to the selected lexemes are returned.

Algorithm 4 MAXVOTE: Voting strategy to select the highest voted lexical entries. See §A.2 for details.

Input: Global lexicon Λ and datapoint-specific sets of lexical entries $\{\lambda^{(1)}, \ldots, \lambda^{(N)}\}$.
Output: Voted set of lexicon entries Λ_{max}.

» Get the set of new lexemes from the N generated sets.
1: $\lambda_\cup \leftarrow \bigcup_{i=1}^{N} \lambda^{(i)}$, $L_{new} \leftarrow \mathcal{L}(\lambda_\cup) \backslash \mathcal{L}(\Lambda)$
» Compute voting dictionary (§A.1).
2: $\mathcal{V} \leftarrow$ AGGREGATEVOTES$(\mathcal{L}(\lambda^{(1)}), \ldots, \mathcal{L}(\lambda^{(N)}))$
» For each string $s \in S(L_{new})$, add to L_{max} the new lexeme with the most votes.
3: $L_{max} \leftarrow \emptyset$
4: **for** each string $s \in S(L_{new})$ **do**
5: $L_{max} \leftarrow L_{max} \cup \{\arg\max_{\ell \in L_{new}} \mathcal{V}_s(\ell)\}$
» Select all the lexical entries with max voted lexemes.
6: $\Lambda_{max} \leftarrow \{e : e \in \lambda_\cup$ and $\mathcal{L}(e) \in L_{max}\}$
7: **return** Λ_{max}

Algorithm 5 CONSENSUSVOTE: Voting strategy to select lexical entries preferred by most datapoints through multiple rounds of voting. See §A.2 for details.

Input: Global lexicon Λ and datapoint-specific sets of lexical entries $\{\lambda^{(1)}, \ldots, \lambda^{(N)}\}$.
Output: Voted set of lexicon entries Λ_{con}.

1: $L^{(i)} \leftarrow \mathcal{L}(\lambda^{(i)}), L_- \leftarrow \emptyset$
2: **repeat**
» For each i, get the set of lexemes that have not been discarded.
3: **for** $i = 1$ to N, $\forall s \in S(L^{(i)})$ **do**
4: $L \leftarrow \{\ell : \ell \in L^{(i)}$ and either $S(\ell) \neq s$ or $\ell \notin L_-\}$
» Replace original set, only if s is still covered.
5: **if** $s \in S(L)$ **then**
6: $L^{(i)} \leftarrow L$
» Compute voting dictionary (§A.1).
7: $\mathcal{V} \leftarrow$ AGGREGATEVOTES$(L^{(1)}, \ldots, L^{(N)})$
8: $L_\cup \leftarrow \bigcup_{i=1}^{N} L^{(i)}$
» Update the set of discarded lexemes L_- with minimally voted lexemes for each string.
9: **for** each string $s \in S(L_\cup)$ **do**
10: $L_- \leftarrow L_- \cup \{\ell : \ell \in L_\cup$ and $\mathcal{V}_s(\ell) = \min_{\ell' \in L_\cup} \mathcal{V}_s(\ell')\}$
11: **until** L_- does not change
» Update lexicon entry sets per datapoint.
12: **for** $i = 1$ to N **do**
13: $\lambda^{(i)} \leftarrow \{e : e \in \lambda^{(i)}$ and $\mathcal{L}(e) \in L^{(i)}\}$
14: $\Lambda_{con} \leftarrow$ MAXVOTE$(\Lambda, \{\lambda^{(1)}, \ldots, \lambda^{(N)}\})$
15: **return** Λ_{con}

Figure 6: Voting strategires pseudo-code.

A.3 Strategy 2: CONSENSUSVOTE

Algorithm 5 provides the pseudocode for the second voting strategy, as described in §4.2. CONSENSUSVOTE iteratively discards lexemes and redistributes their votes between the remaining ones (lines 2-11). This process continues until convergence, i.e., no more lexemes are discarded. To discard lexemes, first we iterate over all datapoint-specific lexeme sets $L^{(i)}$ and the strings s they contain (lines 3-6). For each s and $L^{(i)}$, we create the set L that contains all lexemes in $L^{(i)}$ that are not in the discard set L_-. If L still contains lexemes with the string s, we replace $L^{(i)}$ with it (lines 5-6). This condition is intended to ensure that discarding lexemes will not change the set of sentences that can be parsed. After this update, each $L^{(i)}$ covers the same set of string, but possibly contains fewer lexemes. Next, we update the set of discarded lexemes L_-. First, we use AGGREGATEVOTE to compute the updated voting dictionary (line 7) and collect all lexemes into a single set (line 8). To update L_- we iterate over all strings (lines 9-10), and for each string s add the lexemes with the lowest number of votes to L_-. This process (lines 2-11) repeats until no more entries are discarded. In lines 12-13, the datapoint-specific lexical entry sets are reassigned based on the remaining lexemes in each $L^{(i)}$, and finally, we call MAXVOTE to get the max-voted lexical entries containing the undiscarded lexemes.