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) 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, \( \text{AGGREGATEVOTES}(L_1, \ldots, L_N) \) 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 \( L_{\cup} \rightarrow \mathbb{R} \), which given a lexeme \( \ell \in L_{\cup} \) returns:

\[
\mathcal{V}_s(\ell) = \frac{1}{\sum_{i=1}^{N} \mathbb{1}_{\{\ell \in L_i\}} \mathbb{1}_{\{\ell' \in L_i \land S(\ell') = s\}}}
\]

Where \( \mathbb{1}_{\{\cdot\}} \) is an indicator function. \( \mathcal{V}_s(\ell) \) is the vote assigned to the lexeme \( \ell \) 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 \( L_{\text{new}} \) (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 \( \text{AGGREGATEVOTES} \) (line 2). In lines 4-5, for each string \( s \in S(L_{\text{new}}) \), 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.

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 re-
distributes 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.