**Algorithm 1** Training Algorithm

```
    Input: search results $R$, website descriptions $D$, labels $L$
    Output: updated weights $\mathbf{W}$, biases $\mathbf{b}$
 1: for $iteration < max\ epoch$ do
 2:     initialize weights of encoder $f_{encoder}$ and decoder $f_{decoder}$
 3:     encoder representation $S = f_{encoder}(R)$
 4:     decoder representation $T = f_{decoder}(S)$
 5:     compute Loss $L_1(T, D)$
 6:     input representation to attention classifier $g$: prediction $\hat{L} = g(S)$
 7:     compute Loss $L_2(L, \hat{L})$
 8:     calculate combined loss $argmin(\lambda_1 L_1 + \lambda_2 L_2)$
 9:     update weights $\mathbf{W}$ and biases $\mathbf{b}$ using Adam optimizer
10: end for
```

Line 3-8 is the Attention Classifier.

**Algorithm 2** Predicting Algorithm

```
    Input: search results $R$, trained weights $\mathbf{W}$, biases $\mathbf{b}$
    Output: predicted results $\hat{L}$
 1: build model using trained weights $\mathbf{W}$ and biases $\mathbf{b}$
 2: input $R$ to encoder: representation $S = f_{encoder}(R)$
    Attention Classifier:
 3: feed $S$ through LSTM model: $S' = LSTM(S)$
 4: for for each vector $S'_i$ in $S'$ do
 5:     compute attention $a_i = softmax(S'^T_i q + b)$, where $q, b$ are parameters obtained during training.
 6: end for
 7: calculate attention-weighed representation $S'' = \sum_{i=1}^{n} a_i s'_i$
 8: feed $S''$ through a dense layer with an output dimension of k: $P_k = softmax(MLP(S'')_k)$, where $k$ denotes the index of each category and $P_k$ denotes the predicted probability of each category.
```