Tezan Sahu
170100035
tezansahu@iitb.ac.in

Assignment 1
CS728: Organization of Web Information

Due Date: 07/03/21

Preposition Sense Disambiguation

Following are the implementation details for accomplishing the task of Preposition Sense Disambiguation (by following this paper, as mentioned in Assignment 1.

1 Dataset Preparation

Given the raw XML files containing the sentences & raw sense IDs of the prepositions used in these sentences, we used the approach mentioned below to create the training & validation datasets for the PSD task. The code for this is present in the `create_dataset.py` file.

For each preposition XML file in the `data/Train/Source/` folder, we do the following:

1. Parse the XML as a tree & traverse over each `instance` & do the following:
    - Obtain the `id` of each instance
    - Instead of using the raw `senseid` for each instance, we use definition mappings of The Preposition Project, present in `data/Tratz/Variations/mappings_for_tpp_data` as target labels.
      The exact mapping of `senseid` to the TPP Sense ID is mentioned in the `sense_mapping.json` file
    - For the focus word (i.e. the preposition), we obtain the right & left context vectors (each of size 300) based on k_l & k_r context words left & right of the focus word, as mentioned in the paper. For all our experiments, we use $k_l = l_r = 2$.
    - We us the pre-trained `GoogleNews-vectors-negative300` Word2Vec model for obtaining the embeddings for all words. If a word does not have a pretrained embedding, we use a randomly initialized vector of dimension 300 in its place.
    - In case the number of words in the left context is less than k_l, we use only the number of words available ($< k_l$) to obtain the left context vector. Similar treatment is done for the right context vector as well.
    - We obtain the interplay context vector by applying PCA on the matrix of stacked context word vectors & choosing the 1^{st} principal component (as indicated in the paper).
2. A separate dataframe is created for each preposition, containing the instance `id`, left & right context words, and the context vectors v_l, v_r & v_{inter}.
3. Every such dataframe is split into training & validation sets in the ratio $\sim 80 : 20$.

2 Preposition Sense Disambiguation Algorithms

We implement the following 3 (supervised) approaches to tackle the Preposition Sense Disambiguation task & study the impact of the various tunable hyperparameters, along with ablation of certain context vectors in Section 3.

Note: *For each approach, we train the respective type of multi-class classifier separately for each preposition. The output dimension of each of these classifiers was the number of senses of the corresponding prepositions*

2.1 Weighted k-NN

- **Module used:** `KNeighborsClassifier` from `scikit-learn`
- **Input:** Linear combination of the context vectors ($v = v_l + \beta \cdot v_r + \gamma \cdot v_{inter}$)
- **Tunable Parameters:** β [default $= 1$], γ [default $= 1$], number of neighbors k [default $= 5$]

2.2 MLP (Multi-Layer Perceptron)

- **Module used:** `MLPClassifier` from `scikit-learn` (trained for 200 iterations)
- **Input:** Concatenation of the context vectors (dimension $= 300 \cdot 3 = 900$)
- **Tunable Parameters:** Number of neurons in hidden layer [default $= 100$]

2.3 SVM (Support Vector Machines)

- **Module used:** `SVC` (with *linear kernel*) from `scikit-learn`
- **Input:** Concatenation of the context vectors (dimension $= 300 \cdot 3 = 900$)
- **Tunable Parameters:** Penalty parameter C [default $= 1$]

3 Experiments & Analysis of PSD Algorithms

All the 3 approaches mentioned above have been implemented in the `experiment.py` file. We first perform hyperparameter tuning for the various approaches to maximize the disambiguation accuracy on our validation set. Next, we perform an ablation analysis to understand the importance of the context vectors v_l, v_r & v_{inter}. For the ablation analysis, we choose the best hyperparameters found using the tuning experiments.

Note: *The classification reports of all the models in these experiments can be found in* `./experiments/reports/`

3.1 Hyperparameter Tuning

Hyperparameter tuning experiments are done considering all 3 context vectors (v_l, v_r & v_{inter}) as features.

3.1.1 Weighted k-NN:

# Neighbors (k)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Accuracy	0.716	0.699	0.716	0.722	**0.723**	0.719	0.719	0.717	0.709	0.708

Table 1: Variation of Validation Disambiguation Accuracy with change in k (keeping $\beta = \gamma = 1$)

		γ			
		0.1	**0.5**	**1**	**5**
β	**0.1**	**0.758**	0.743	0.731	0.615
	0.5	0.757	0.755	0.733	0.637
	1	0.711	0.722	0.723	0.643
	5	0.564	0.570	0.573	0.630

Table 2: Variation of Validation Disambiguation Accuracy with change β & γ (keeping $k = 5$)

3.1.2 Multi-Layer Perceptron (MLP)

# Neurons in Hidden Layer	**10**	**50**	**100**	**200**
Accuracy	0.796	0.819	**0.820**	**0.820**

Table 3: Variation of Validation Disambiguation Accuracy with change in number of neurons in hidden layer

3.1.3 Support Vector Machines (SVM)

Penalty Parameter (C)	**0.01**	**0.05**	**0.1**	**0.5**	**1**	**5**	**10**
Accuracy	0.599	0.753	0.809	**0.829**	0.828	0.827	0.826

Table 4: Variation of Validation Disambiguation Accuracy with change in penalty parameter C

3.2 Ablation Analysis

To understand the importance of the context feature vectors in the disambiguation task, we do an ablation analysis by training the models mentioned above with the same hyperparameters that achieved highest accuracies previously, but this time removing one of the context vectors. The results of testing these models on the validation dataset are given in Table 5.

Feature Type	(l, r)	(l, i)	(r, i)	(l, r, i)
Weighted k-NN	0.760	0.754	0.600	0.758
MLP	0.826	0.792	0.653	0.820
SVM	**0.835**	0.805	0.654	0.829

Table 5: Performance of the supervised PSD models. (l, i), (l, r) and (r, i) correspond to feature ablation results

Through the above analysis, the left context feature appears to be the most beneficial (removing it causes a drastic decrease in the accuracy of the models). Moreover, for the given training dataset, removing the context-interplay vector actually helps in boosting the performance of all the 3 models.

4 Testing

We use the model that achieved the highest accuracy in the PSD task on our validation dataset (**SVM with $C = 0.5$ & using only v_l & v_r**) to obtain the predictions for the test data. The final outputs can be found in the `test_out/` folder.

5 Sequence of Commands Used to Obtain Outputs

Following is the sequence of commands that can be used to create the required datasets, train/validate the model & predict the labels for the test data:

Create the Training, Validation & Test Datasets containing the Context Vectors:

```
python create_datasets.py --dataDir ./data/Train/Source --outDir ./datasets --kl 2 --kr 2
python create_datasets.py --dataDir ./test_out --outDir ./datasets --test --kl 2 --kr 2
```

Train the SVM Model with $C = 0.5$ using v_l & v_r:

```
python experiment.py --modelType svm --C 0.5 --features lr
```

Use the Trained Model to Predict Labels for the Test Dataset:

```
python experiment.py --test --modelsDir ./experiments/models/lr_svm_C=0.5 --features lr
--modelType svm --C 0.5
```