Lab Class NLP: Text Models II

Submit your solutions **until 29.05.2024, 23:59 pm** on Moodle. The submission should include a single PDF file with your group's solutions to the exercises.

Exercise 1 : Levenshtein Distance (1+4=5 Points)

The lecture introduced the Levenshtein distance algorithm as a method for measuring the similarity between two strings.

(a) Briefly explain how the Levenshtein distance algorithm works.

(b) Word Error Rate (WER) is the word-level Levenshtein distance metric that is used to evaluate the performance of machine translation systems. WER calculates the minimum number of word-level edits (insertions, deletions, or substitutions) required to change one sentence into another.

Suppose that you are designing a machine translation system. You have the following input sentence:

Er hätte der Berichterstattung gegenber dem Parlament Vorrang einräumen müssen .

A human translator has provided you with the following reference sentence:

He should have prioritised giving that report to Parliament .

A machine translation system produces the following output:

He should have the reporting to the Parliament to give priority .

(b1) Calculate the WER between the machine translation output and the human translation in the example above. Assume that the cost of each operation (insertion, deletion, substitution) is 1. Include a table that shows the distance matrix for the two sentences. Report the resulting WER value.

(b2) Illustrate at least one advantage and one limitation of the WER metric for evaluating machine translation systems using the example above.

Exercise 2 : Word Vectors (1+1+1+1=4 Points)

The lecture introduced distributional representations of words (word vectors) as embeddings of words in a latent space.

(a) Which of the following statements about word vectors are true? (Select all that apply)

- ☐ Word vectors are typically learned from large text corpora using unsupervised learning methods.
- ☐ Only words can be embedded as word vectors, not subword units.
- ☐ Word vectors are used to represent words as dense vectors in a low-dimensional space.
- ☐ Word vectors are used to represent words as sparse vectors in a high-dimensional space.
- ☐ Word vectors can capture semantic relationships between words.
- ☐ The higher the dimensionality of the word vectors, the more information they can capture.

(b) What is distributional hypothesis in the context of word vectors?

(c) Which of the following equations should hold for good word embeddings? (Select all that apply)

- ☐ $\mathbf{w}_{\text{boy}} - \mathbf{w}_{\text{girl}} \approx \mathbf{w}_{\text{brother}} - \mathbf{w}_{\text{sister}}$
- ☐ $\mathbf{w}_{\text{cat}} - \mathbf{w}_{\text{dog}} \approx \mathbf{w}_{\text{puppy}} - \mathbf{w}_{\text{kitten}}$
- ☐ $\mathbf{w}_{\text{berlin}} - \mathbf{w}_{\text{tokyo}} + \mathbf{w}_{\text{japan}} \approx \mathbf{w}_{\text{germany}}$
- ☐ $\mathbf{w}_{\text{walk}} - \mathbf{w}_{\text{walked}} - \mathbf{w}_{\text{buy}} \approx \mathbf{w}_{\text{bought}}$

(d) What is the difference between static and contextualized word vectors? Give at least one example for each type of word vector.

© WIEGMANN/MIRZAKHMEDOVA/KOLYADA 2024

Exercise 3 : Word Mover Distance (1+1+1=3 Points)

The lecture introduced the Word Mover Distance (WMD) for measuring word vector similarity. WMD finds the minimum cumulative transportation cost to move all words from one sentence to words in another sentence in an embedding space.

You are given the sentences A, B, and C and the 3-dimensional word vectors for all occurring words:

- Sentence A: "The **`cat`** **`climbed`** the tree."
- Sentence B: "The **`feline`** **`scaled`** the tree."
- Sentence C: "The **`kitten`** **`ascended`** the tree."

	`the`	`cat`	`climbed`	`tree`	`feline`	`scaled`	`kitten`	`ascended`
d_1	0.1	0.4	0.7	1.0	0.35	0.75	0.45	0.77
d_2	0.2	0.5	0.8	1.1	0.55	0.85	0.57	0.87
d_3	0.3	0.6	0.9	1.2	0.65	0.95	0.67	0.97

(a) Calculate the WMD between Sentence A and Sentence B.

(b) Calculate the WMD between Sentence A and Sentence C.

(c) Which sentence B or C is more similar to Sentence A?

© WIEGMANN/MIRZAKHMEDOVA/KOLYADA 2024

Exercise 4 : Sentence Embeddings (1+1+0+2=4 Points)

The lecture introduced sentence embeddings as a way to represent sentences in a continuous vector space. One approach to generating sentence embeddings is to average the word embeddings $\mathbf{w}$ of all words in a sentence s:

$$\mathbf{s}_{emb} = \frac{1}{|s|} \sum_{\mathbf{w}_i \in s} \mathbf{w}_i$$

Given the same sentences A, B, and C and the 3-dimensional word vectors for all occurring words from the Exercise 2:

(a) Calculate the sentence embeddings for Sentence A, B, and C using vector averaging.

(b) What are the limitations of the vector averaging approach for generating sentence embeddings? Name at least two.

(c) What are other methods for generating sentence embeddings? Name at least two and briefly describe them.

(d) One approach to measuring the similarity between two sentences is to calculate the cosine similarity between their sentence embeddings:

$$\mathit{sim}_{cosine}(\mathbf{s}_1, \mathbf{s}_2) = \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{||\mathbf{s}_1|| \cdot ||\mathbf{s}_2||}$$

(d1) Interpret the following cosine similarity values between two sentence embeddings:

a) $\mathit{sim}_{cosine}(\mathbf{s}_1, \mathbf{s}_2) = -1$
b) $\mathit{sim}_{cosine}(\mathbf{s}_1, \mathbf{s}_2) = 0$
c) $\mathit{sim}_{cosine}(\mathbf{s}_1, \mathbf{s}_2) = 1$

(d2) Calculate the cosine similarity between Sentence A and Sentence B, and between Sentence A and Sentence C. Which sentence is more similar to Sentence A according to the cosine similarity measure?

© WIEGMANN/MIRZAKHMEDOVA/KOLYADA 2024

Exercise 5 : Shared Task – Paraphrase Identification $\boxed{\text{P}}$ (4 Points)

Task: Given a piece two sentences, determine whether they are paraphrases of each other.

Dataset: The dataset consists of two files: `text.jsonl` and `labels.jsonl`. The text file contains pairs of sentences, while the labels file contains the ground truth for each pair.

Each dataset (train, validation, test) is split in two files:

(a) The `text.jsonl` contains the text data with the following schema:
```
{"id": 0,
 "sentence1": "Amrozi accused his brother , whom he called “ the witness ” , of
deliberately distorting his evidence .",
 "sentence2": "Referring to him as only “ the witness ” , Amrozi accused his
brother of deliberately distorting his evidence ."}
```
(b) The `labels.jsonl` contains the ground truth (0 or 1) for each pair with the following schema:
```
{"id": 0, "label": 1}
```

Task: Develop a program that, without human intervention, determines whether the pair of sentences are paraphrases. Write all predictions into a new file `predictions.jsonl` following the schema of the labels file given above. Your truth file must contain a prediction for all the examples in the dataset and should preserve the examples order.

Evaluation: The evaluation metric is Matthews correlation coefficient (MCC). To get points for this task, you must beat the provided baseline performance on the *validation* set. You can find the performance of the baseline on the task leaderboard. You can find the code for the baseline solution that employs Levenshtein distance on your group GitHub repository in the directory: `paraphrase-identification`.

Submission: Create a directory `paraphrase-identification-submission` in your group GitHub repository. Include the following files:

1. `Dockerfile` – to build your environment with all dependencies and run your code;
2. `train.py`* – script that trains your model on the train set and saves it;
3. `model.joblib`* – a file with the trained model's weights;
4. `run.py` – script that reads the input data, loads the model (if training is involved), makes predictions, and writes them to the output file.

Submit your solution to tira.io for evaluation†. Baseline solution, as well as instructions, examples and for submitting your predictions can be found in your GitHub repository assigned by TIRA. More details in the step-by-step guide [here]. You can submit your solution multiple times. The best submission will be used for grading.

Include the following information in your PDF submission:

(a) Your team name on TIRA.

(b) A brief description of your approach.

*Only applicable when your approach is based on a learning algorithm (Logistic regression, Naive Bayes, Neural Networks, etc). For an example, see authorship-verification-bayes directory. These files are not necessary in cases when your approach is based on heuristics (e.g. see authorship-verification-trivial)

†No training can be done on TIRA, so make sure to train your model beforehand.

© WIEGMANN/MIRZAKHMEDOVA/KOLYADA 2024