Master on Artificial Intelligence Universitat Politècnica de Catalunya

Advanced Human Language Technologies

Final Exam June 15^{th}, 2023

Exercise 1. Estimation & Smoothing

We want to build yet another opinion mining system about movie reviews, and our first requirement is to build a language model.
We have the following training sentences:

- *The movie was amazing*
- *I did not like the plot*
- *The ending was surprising*
- *The actor was boring*
- *I did like the movie*

We estimate that the size of the vocabulary used globally in the movie reviews will be of 5,000 words.

1. Compute the following probabilities of a bigram language model for the movie reviews, both using MLE and Lidstone Law with $\lambda = 0.1$:

   - $P(amazing|was)$
   - $P(like|did)$
   - $P(awful|was)$
   - $P(movie|the)$
   - $P(was|ending)$

   Justify your answer and the values chosen for B, N, and N_0 where it applies.

2. Compute the most likely next 2 words to complete the sentence: *The actor did like ...*.
   Justify your answer.

3. Use the MLE language model resulting from the above training data to compute the probability of the following sentences

   - *The ending was amazing*
   - *I did like the actor*

Develop your computations, do not provide just a numeric result.
You may leave probabilities as fractions.

SOLUTION

1. The requested probabilities are:

	MLE	LID$_{\lambda=0.1}$
$P(amazing\|was)$	$\frac{count(\text{was amazing})}{count(\text{was})} = \frac{1}{3}$	$\frac{count(\text{was amazing})+\lambda}{count(\text{was})+B\lambda} = \frac{1+0.1}{3+5{,}000\cdot 0.1} = \frac{1.1}{503}$
$P(like\|did)$	$\frac{count(\text{did like})}{count(\text{did})} = \frac{1}{2}$	$\frac{count(\text{did like})+\lambda}{count(\text{did})+B\lambda} = \frac{1+0.1}{2+5{,}000\cdot 0.1} = \frac{1.1}{502}$
$P(awful\|was)$	$\frac{count(\text{was awful})}{count(\text{was})} = \frac{0}{3}$	$\frac{count(\text{was awful})+\lambda}{count(\text{was})+B\lambda} = \frac{0+0.1}{3+5{,}000\cdot 0.1} = \frac{0.1}{503}$
$P(movie\|the)$	$\frac{count(\text{the movie})}{count(\text{the})} = \frac{2}{5}$	$\frac{count(\text{the movie})+\lambda}{count(\text{the})+B\lambda} = \frac{2+0.1}{5+5{,}000\cdot 0.1} = \frac{2.1}{505}$
$P(was\|ending)$	$\frac{count(\text{ending was})}{count(\text{ending})} = \frac{1}{1}$	$\frac{count(\text{ending was})+\lambda}{count(\text{ending})+B\lambda} = \frac{1+0.1}{1+5{,}000\cdot 0.1} = \frac{1.1}{501}$

   N is the number of observations of the conditioning word in each case (i.e. the unigram count). $B = 5,000$ in all cases, since this is the number of potentially observable words in our data, and thus, the number of potential bigram continuations after any word.

2. Since the language model is a bigram model, the next word will depend only on the last one, in this case *like*.

   Word *like* happens twice in the training data (sentences 2 and 5). In both cases, it is followed by *the*. So, with MLE we have $P(the|like) = 2/2 = 1.0$, and with Lidstone we have $P(the|like) = (2 + 0.1)/(2 + 5000 \cdot 0.1)$, but in both cases the most likely next word is *the*.

   So we get: *The actor did like the...*

   The second word to be added will be the most likely to follow *the* (again, the bigram model only considers the last word to generate the next one).

   Word *the* happens 5 times in the training data (once in each sentence). It is followed by *movie* twice and once by each of *plot*, *ending*, and *actor*. So the most likely word after *the* is *movie*, thus the sentence after generating the next two words is:

   The actor did like the movie

3.

$$\begin{aligned} P(\textit{The ending was amazing}) =& \pi(the) \times P(ending|the) \times P(was|ending) \times P(amazing|was) \\ =& \frac{3}{5} \times \frac{1}{5} \times \frac{1}{1} \times \frac{1}{3} = \frac{1}{25} \end{aligned}$$

$$\begin{aligned} P(\textit{I did like the actor}) =& \pi(I) \times P(did|I) \times P(like|did) \times P(the|like) \times P(actor|the) \\ =& \frac{2}{5} \times \frac{2}{2} \times \frac{1}{2} \times \frac{2}{2} \times \frac{1}{5} = \frac{1}{25} \end{aligned}$$

Exercise 2. Features

We want to provide our movie review opinion mining system with the ability to spot opinions about specific movie characteristics (e.g. a user may like the actors but dislike the ending).

For this, we model a sequence labeling task to identify which words indicate a positive or negative opinion.

We have the following data:

$\mathcal{X}$	The	actors	were	bad	but	I	liked	the	ending
$\mathcal{Y}$	O	O	B-neg	I-neg	O	O	B-pos	O	O
$\mathcal{X}$	The	movie	was	good	but	I	was	disappointed	
$\mathcal{Y}$	O	O	B-pos	I-pos	O	O	B-neg	I-neg	
$\mathcal{X}$	I	did	not	like	it	,	a	boring	movie
$\mathcal{Y}$	O	B-neg	I-neg	I-neg	O	O	O	B-neg	I-neg

And the following feature templates:

$$\mathbf{f}_{1,a,l}(\mathcal{X}, i, t) = \begin{cases} 1 & \text{if } w_i = a \wedge w_{i-1} \in \{\textit{was}, \textit{were}\} \wedge t = l \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_{2,a,l}(\mathcal{X}, i, t) = \begin{cases} 1 & \text{if } w_i \in \{\textit{like}, \textit{liked}\} \wedge w_{i-1} = a \wedge t = l \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_{3,l}(\mathcal{X}, i, t) = \begin{cases} 1 & \text{if } w_i \in \{\textit{bad}, \textit{awful}, \textit{boring}, \textit{disappointed}\} \wedge t = l \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{f}_{4,l}(\mathcal{X}, i, t) = \begin{cases} 1 & \text{if } w_i \in \{\textit{the}, \textit{that}, \textit{some}, \textit{a}\} \wedge t = l \\ 0 & \text{otherwise} \end{cases}$$

1. (a) Assuming a vocabulary size $|\mathcal{V}| = 5,000$, which is the dimension of the potential feature space these templates can create? Justify your answer.

   (b) Which is the dimension of the feature space actually instantiated by these templates when applied on the three-sentence training dataset above?

2. Given the following test sentence $\mathcal{X}$ and hypothesis tag sequence $\mathcal{Y}$:

$\mathcal{X}$	Bad	movie	,	the	ending	was	awful	but	I	liked	the	actors
$\mathcal{Y}$	B-neg	O	O	O	O	B-neg	I-neg	O	O	B-pos	O	O

   compute the feature vectors $\mathbf{f}(\mathcal{X}, i, t)$ for each position i, and the global feature vector $\mathbf{f}(\mathcal{X}, \mathcal{Y})$. Highlight which features in the global vector that are present in the vector space instantiated by the three training sentences above.

SOLUTION

1.

(a)
- Template $\mathbf{f}_{1,a,l}$ can potentially be instantiated with every possible value for $a \in \mathcal{V}$ combined with every possible value for $l \in \{\texttt{O}, \texttt{B-pos}, \texttt{I-pos}, \texttt{B-neg}, \texttt{I-neg}\}$, that is $5,000 \times 5 = 25,000$ potential features
- The same rational can be applied to $\mathbf{f}_{2,a,l}$, so it may potentially generate another $25,000$ features
- Templates $\mathbf{f}_{3,l}$ and $\mathbf{f}_{4,l}$ can potentitally be instantiated for each value of l, producing up to 5 features each.

So the potential dimension of the feature space is $25,000 + 25,000 + 5 + 5 = 50,010$

(b)
- Template $\mathbf{f}_{1,a,l}$ is instantiated three times (one as $\mathbf{f}_{1,\text{bad},\texttt{I-neg}}$ in the first sentence, and two more as $\mathbf{f}_{1,\text{good},\texttt{I-pos}}$, $\mathbf{f}_{1,\text{disappointed},\texttt{I-neg}}$ in the second)
- Template $\mathbf{f}_{2,a,l}$ is instantiated twice (one as $\mathbf{f}_{2,\text{I},\texttt{B-pos}}$ in the first sentence, another as $\mathbf{f}_{2,\text{not},\texttt{I-neg}}$ in the third)
- Template $\mathbf{f}_{3,l}$ is instantiated three times, one in each sentence, as: $\mathbf{f}_{3,\texttt{I-neg}}$, $\mathbf{f}_{3,\texttt{I-neg}}$, and $\mathbf{f}_{3,\texttt{B-neg}}$ respectively. Since the instantiations in the first two sentences produce the same feature, this template produces only two new features.
- Template $\mathbf{f}_{4,l}$ is instantiated four times: Two in the first sentence as $\mathbf{f}_{4,\texttt{O}}$ by the two ocurrences of `the`. Second and third sentences contain respectively an ocurrence of `the`, and one of `a`, which produce the same feature $\mathbf{f}_{4,\texttt{O}}$. So, only one feature instance is produced by this template on this dataset.

So, the dimension of the feature space instantiated by this dataset is $3 + 2 + 2 + 1 = 8$.

2.

i	feature vector $\mathbf{f}(\mathcal{X}, i, t)$
1	$^*\mathbf{f}_{3,\texttt{B-neg}}$
2	-
3	-
4	$^*\mathbf{f}_{4,\texttt{O}}$
5	-
6	-
7	$\mathbf{f}_{1,\text{awful},\texttt{I-neg}}$, $^*\mathbf{f}_{3,\texttt{I-neg}}$
8	-
9	-
10	$^*\mathbf{f}_{2,\text{I},\texttt{B-pos}}$
11	$^*\mathbf{f}_{4,\texttt{O}}$
12	-

* Feature present in the space instantiated by the training data

Global vector $\mathbf{f}(\mathcal{X}, \mathcal{Y})$:

feature	value
$\mathbf{f}_{1,\text{awful},\texttt{I-neg}}$	1
$\mathbf{f}_{2,\text{I},\texttt{B-pos}}$	1
$\mathbf{f}_{3,\texttt{B-neg}}$	1
$\mathbf{f}_{3,\texttt{I-neg}}$	1
$\mathbf{f}_{4,\texttt{O}}$	2

Exercise 3. Parsing

One user in out platform wrote the review:

It was a huge bomb explosion movie

1. Draw dependency trees for the following interpretations:
   (a) The movie was huge and it was about explosions. The movie was also about bombs.
   (b) The movie was huge and contained the explosion of a bomb.
   (c) A huge bomb exploded in the movie.
   (d) The movie was about explosions, and also about a huge bomb.

2. Given the following emulation of the behaviour of a transition dependency parser with an arc-standard model (i.e. with operations *shift*, *left-arc*, and *right-arc* between the two topmost stack elements), answer the questions below:

Stack	Buffer	Edges	Transition
*	It was a huge bomb explosion movie	{}	shift
* It	was a huge bomb explosion movie	{}	shift
* It was	a huge bomb explosion movie	{}	left-arc
* **was**	a huge bomb explosion movie	{**(2,1)**}	shift
* was a	huge bomb explosion movie	{(2,1)}	shift
* was a huge	bomb explosion movie	{(2,1)}	shift
* was a huge bomb	explosion movie	{(2,1)}	left-arc
* was a **bomb**	explosion movie	{(2,1), **(5,4)**}	shift
* was a bomb explosion	movie	{(2,1), (5,4)}	???

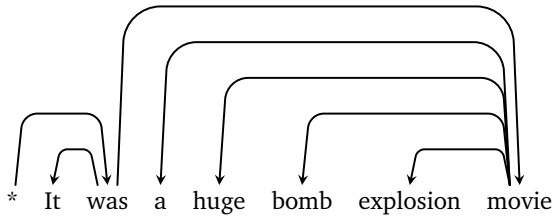
   (a) If we apply *left-arc* as the next transition
      - Which arc would be added to the tree?
      - How many of the four interpretations above are still possible, and which ones? Justify your answer.

   (b) If we apply *shift* as the next transition instead
      - How many of the four interpretations above are still possible, and which ones? Justify your answer.
      - Which should be the following transitions to complete the tree and which arcs would each transition add?

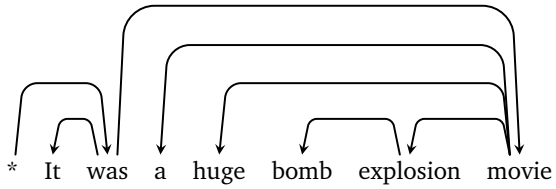
SOLUTION

1.

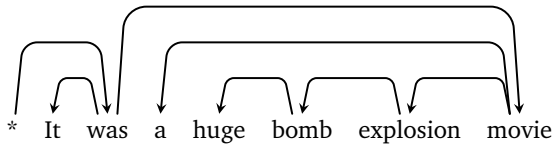
(a) The movie was huge and it was about explosions. The movie was also about bombs.



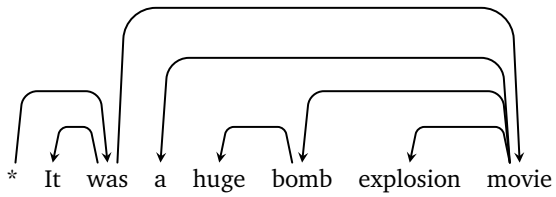
(b) The movie was huge and contained the explosion of a bomb.



(c) A huge bomb exploded in the movie.



(d) The movie was about explosions, and also about a huge bomb.



(a) If we apply *left-arc* as the next transition

- A *left-arc* transition will add an arc between the top two words in the stack, putting the leftmost under the rightmost, since the top two words in the stack are *bomb* and *explosion*, this will put *bomb* under *explosion*, i.e. adding arc $(6, 5)$
- After adding this arc, only interpretations that contain all the arcs created so far can be reached. Only one interpretation (c) remains, since it is the only containing arcs $(5, 4)$ and $(6, 5)$

(b) If we apply *shift* as the next transition instead

- A *shift* transition will move *movie* to the stack, skipping the addition of arc $(6, 5)$ (if we want to add $(6, 5)$ later, we will need to add $(6, 7)$ first, which would discard all 4 interpretations). Thus, the only remaining possibilities will be those containing arc $(5, 4)$ and **not** containing $(6, 5)$. The only interpretation satisfying these constraints is (d) .
- Next steps will be: *left-arc* will add edge $(7, 6)$, *left-arc* will add edge $(7, 5)$, *left-arc* will add edge $(7, 3)$, *right-arc* will add edge $(2, 7)$, *right-arc* will add edge $(0, 2)$, *stop*.

Exercise 4. Word Embeddings

One possible approach to generate distributional word embeddings from a large corpus is to use the word co-occurrence matrix and then apply a dimensionality reduction algorithm. Describe the process of generating such embeddings, highlighting the possible issues and solutions with this approach. What are GloVe embeddings and how do they compare to this approach?

SOLUTION

1. Building the word co-occurrence matrix:

   - Iterate over the corpus and count the co-occurrence of words within a specific window size. Each cell in the matrix represents the co-occurrence count of two words.
   - Optionally, you can apply pre-processing steps like lowercasing, removing punctuation, and tokenizing the text.

   Issues and solutions:

   - **Issue**: The co-occurrence matrix can be very large and sparse, making it computationally expensive to store and process.
   - **Solution**: Use sparse matrix data structures to efficiently store the matrix and optimize memory usage.
   - **Issue**: Selecting an appropriate context window size is crucial. A small window may not capture enough semantic information, while a large window may introduce noise and unrelated co-occurrences.
   - **Solution**: Experiment with different window sizes to find a balance between capturing local context and avoiding noise.

2. Applying pointwise mutual information (PMI) or positive pointwise mutual information (PPMI):

   - Calculate the PMI or PPMI values for each co-occurrence count in the matrix.
   - PMI measures the statistical dependence between two words, while PPMI handles negative values and focuses on the positive associations.

   Issues and solutions:

   - **Issue**: The co-occurrence matrix may contain many zero entries, resulting in undefined or infinite PMI/PPMI values.
   - **Solution**: Apply some form of smoothing or regularization to handle the zero counts. One common approach is adding a small constant to each count (e.g., 1).
   - **Issue**: Common words that do not carry much information might have a big impact in the co-occurence matrix.
   - **Solution**: Filter or lower the weight of such words:
     - Term Frequency-Inverse Document Frequency (TF-IDF): You can weight the co-occurrence counts based on the TF-IDF metric to give more importance to rare words and filter out common stopwords that don't carry much semantic information.
     - Stopwords: Exclude common words (e.g., "and," "the," "is") from the co-occurrence matrix to reduce noise and focus on more meaningful word relationships.

3. Applying dimensionality reduction (e.g., SVD):

   - Perform dimensionality reduction on the PMI/PPMI matrix to obtain lower-dimensional word representations.
   - Singular Value Decomposition (SVD) is a popular method for this purpose, which decomposes the matrix into three matrices: U, Σ, and V, where U and V represent orthogonal bases and Σ contains singular values.

Issues and solutions:

- **Issue**: The SVD operation can be computationally expensive, especially for large matrices.
- **Solution**: Utilize optimized implementations of SVD or consider approximations like truncated SVD or randomized SVD to reduce the computational complexity.

* GloVe: GloVe (Global Vectors for Word Representation) embeddings are a type of distributional word embeddings that combine global word co-occurrence statistics with local context windows.

  - From the co-occurrence matrix, GloVe computes the probabilities of two words co-occurring together. These probabilities are influenced by both global and local word relationships.
  - GloVe embeddings are obtained by performing a factorization of the word co-occurrence matrix using the learned word vectors. The factorization-based approach of GloVe is computationally more efficient than techniques like SVD applied to the co-occurrence matrix.
  - GloVe introduces an objective function that aims to learn word vectors that encode semantic relationships by minimizing the difference between the dot product of word vectors and the logarithm of the co-occurrence probabilities.

Exercise 5. Neural Networks and Transformers

Compare, including their strengths and weakenesses, the following pre-training approaches for contextual embeddings: causal LM, bidirectional LM and masked LM. Can you name a model in which each one of them is used?

SOLUTION

Causal Language Model (CLM):

- In a CLM, the model predicts the next word in a sequence based on the preceding words. It can only attend to the left context, making it unidirectional.
- **Strengths:**
  - Preserves the autoregressive property, ensuring that the model generates text sequentially and follows a coherent flow.
  - Effective in capturing long-range dependencies in the left context.
- **Weaknesses:**
  - Limited in incorporating information from the right context, which can limit its ability to understand the overall meaning of a sentence.
- **Example model:** GPT (Generative Pre-trained Transformer)

Bidirectional Language Model (BiLM):

- A BiLM processes the entire input sequence in both forward and backward directions. It considers both left and right contexts to predict each word.
- **Strengths:**
  - Captures dependencies from both directions, allowing for a more comprehensive understanding of the context.
  - Exhibits better contextual awareness and is capable of modeling complex interactions between words.
- **Weaknesses:**
  - Does not preserve the autoregressive property, as it uses future information during pre-training.
  - May introduce potential information leakage from the future in downstream tasks.
- **Example model:** ELMo (Embeddings from Language Models)

Masked Language Model (MLM):

- An MLM pre-training approach masks random words in the input sequence and requires the model to predict those masked words based on the remaining context.
- **Strengths:**
  - Allows bidirectional training by leveraging both left and right contexts.
  - Enables the model to learn robust contextual representations by filling in the missing words.
- **Weaknesses:**
  - Introduces a mismatch between pre-training and fine-tuning, as the model does not encounter masked words during fine-tuning.
  - May result in biased representations if the masking strategy is not carefully designed.
- **Example model:** BERT (Bidirectional Encoder Representations from Transformers)

Exercise 6. LLM

1. What is content hallucination in LLM? How does it relate to biases in LLM? Can it be prevented? How?
2. What are zero-shot, one-shot, and few-shot tasks in the context of language models and NLP? Include an example of each. How was GPT-3 trained to achieve great performance in these tasks?

SOLUTION

1. Content hallucination in LLMs (Large Language Models) is a phenomenon where the model generates text that is incorrect, nonsensical, or not real. It occurs because the model's primary objective is to generate text that is coherent and contextually appropriate, rather than factually accurate. The model's training data may contain inaccuracies, inconsistencies, and fictional content, and the model has no way of distinguishing between fact and fiction.

   Content hallucination in LLMs can relate to biases in LLMs in several ways. One way is that hallucinations can reflect the biases present in the training data, such as stereotypes, prejudices, or misinformation. Another way is that hallucinations can introduce new biases that are not present in the training data, such as implausible scenarios, unsupported claims, or factual errors. These biases can affect the quality and reliability of the LLM-generated outputs and potentially harm the users or applications that rely on them.

   Content hallucination in LLMs can be prevented or mitigated by using various methods, such as:

   - Data cleaning and filtering: Removing or correcting noisy, inaccurate, or harmful data from the training corpus.
   - Data augmentation and diversification: Adding more diverse and representative data to the training corpus to reduce overfitting and improve generalization.
   - Model regularization and fine-tuning: Applying techniques such as dropout, weight decay, or adversarial training to reduce the model's complexity and sensitivity to spurious correlations.
   - Model evaluation and testing: Developing rigorous and comprehensive metrics and tests to measure model performance and detect hallucinations across different domains and tasks.
   - Model explanation and verification: Using methods such as attention mechanisms, saliency maps, or counterfactual analysis to understand how the model generates outputs and verify their validity.
   - Model feedback and correction, such as RLHF (Reinforcement Learning with Human Feedback): Incorporating human feedback or external knowledge sources to correct or improve the model's outputs. The model learns to generate outputs that are more aligned with human preferences and expectations, and avoid outputs that are hallucinated or biased.

2. - Zero-shot tasks: Zero-shot tasks involve performing a task without any specific training examples or explicit fine-tuning on that task. The model is expected to generalize from its pre-trained knowledge and apply it to the new task based on high-level instructions or prompts.
   - One-shot tasks: One-shot tasks involve performing a task with only a single or very few examples provided during training. The model needs to leverage its pre-existing knowledge to generalize from the limited input data and perform the task effectively.
   - Few-shot tasks: Few-shot tasks involve performing a task with a small number of training examples, typically more than one but significantly fewer than what would be required for traditional training. The model should be able to learn from this limited data and generalize to new examples.
   - GPT-2 and GPT-3 were trained using synthetic examples in addition to crawled text. The use of synthetic examples was crucial for GPT-3 to achieve its impressive performance in zero-shot and few-shot scenarios. By combining prompts and target outputs from different tasks, the model could learn to generalize across tasks and generate appropriate responses even for tasks it had not explicitly seen during training. This approach demonstrated the model's ability to leverage its pre-existing knowledge and adapt it to new tasks with minimal training data.