Natural Language Understanding, Generation and Machine Translation - Week 1 - Introduction to Machine Translation

Antonio León Villares

January 2023

Contents

Based on a Tutorial by Adam Lopez.

1 Conditional Language Modelling

1.1 General Language Models

- **What is a language model?**
  - a **probabilistic model** for **strings**
  - for example, we can train a model for **headline generation**
- **What are conditional language models?**
  - **language models**, where language prediction is **conditioned** on some input
  - for example:
    - *speech recognition* (conditioned on **speech signal**)
    - *machine translation* (conditioned on text in another language)
    - *text completion*: (conditioned on first words of a sentence)
    - *OCR* (conditioned on **images** of text)
    - *image captioning* (conditioned on an image)
    - *grammar checking* (conditioned on surrounding words)
- **How can language models be interpreted as functions?**
  - we consider a **finite vocabulary** V
  - a **language model** can be thought of as a function:

$$P : V^* \to [0, 1]$$

    where V^* denotes the set of word sequences (or arbitrary length) which can be constructed from V
  - we must ensure that all the probabilities outputted by P add up to 1
  - this defines a **probability distribution**, whose random variables can be, for example, words at a given position in a sentence (i.e w_1 is the RV representing the first word in the sentence provided)

1.2 The N-Gram Language Model

- **How can we define the probability of a sentence?**
  - consider a sentence represented by $\underline{w}$ (such that w_i represents the ith word in $\underline{w}$), and assume that $|\underline{w}| = L$
  - using the **chain rule** of probability, we can define:

$$\begin{aligned} P(\underline{w}) &= P(w_1, \ldots, w_L) \\ &\equiv P(w_{1:L}) \\ &= P(w_1) \times P(w_2 \mid w_1) \times \ldots \times P(\texttt{</s>} \mid w_{1:L}) \\ &= \prod_{i=1}^{L+1} P(w_i \mid w_{1:i-1}) \end{aligned}$$

- **Why is this representation for a sentence probability?**
  - there are (potentially) no limitations for $|\underline{v}| = L$

- this model might thus rely on (potentially) **infinite histories**

- **How do n-gram models deal with infinite histories?**
  - instead of conditioning on the **whole** history, we use a **Markov assumption**, and consider on a **fixed history**
  - for an **n-gram**, we consider windows of width n, such that the nth word is conditioned on the $n-1$ previous words:

$$\forall i \in [1, L+1],\ P(w_i \mid w_{1:i-1}) \approx P(w_i \mid w_{i-n+1:i-1})$$

- **How can n-gram probabilities be estimated?**
  - we can use **Maximum Likelihood Estimation**: given a corpus of word occurrences, we can use **counts** to estimate n-gram probabilities:

$$P(w_2 \mid w_1) = \frac{Count(w_1, w_2)}{Count(w_1)} \qquad P(w_3 \mid w_1, w_2) = \frac{Count(w_1, w_2, w_3)}{Count(w_1, w_2)}$$

  - such a model would **maximise** the **likelihood function**: given some training data $\mathcal{D}$, this is a function mapping models θ to a probability
  - for example, for bigrams:

$$P(\mathcal{D} \mid \theta) = \prod_{w_1, w_2 \in V^2} P(w_2 \mid w_1, \theta)$$

    and the MLE estimation is a setting $\hat{\theta}$, such that:

$$\hat{\theta} = \arg\max_{\theta} P(\mathcal{D} \mid \theta)$$

- **What are the main issues with n-grams?**
  - **Data Sparsity**: for good models, we require large n, but this can lead to **data sparsity**: high-order n-grams will barely appear, so MLE estimates will mostly be 0 (these could be **structural zeroes** (the n-grams are never produced by the language) or **sampling zeroes** (we haven't yet found them in training))
  - **Model Size**: to get a good model, we'd require **billions** of word sequences, which requires a lot of **memory**

- **How can n-grams be used to generate text?**
  - if we have a word sequence $w_{1:k}$, we can predict the next word via:

$$\hat{w}_{k+1} = \arg\max_{w_{k+1}} P(w_{k+1} \mid w_{1:k})$$

  - this is particularly useful when processing inputs in real-time (word-by-word)

2 Machine Translation With N-Grams

> ***Machine Translation*** *involves converting an input* $\underline{x}$ *(written in language A) into an output* $\underline{y}$ *(written in language B), such that the* ***meaning*** *is preserved*

2.1 Defining the MT Model

- **What is the main hurdle in MT?**

- **sentence length** can **vary**, since some words might not have a direct translation, or might be included just for structure
- for example:

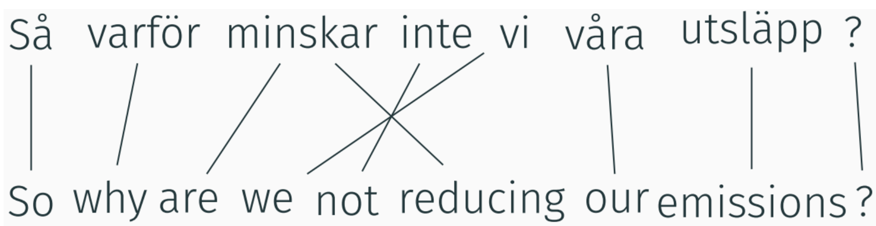
"Me gusta jugar al fútbol"
"I like playing football"

here, the article "al" appears before the noun, which would sound weird in English ("I like playing the football")

- these nuances can become even more pronounced when language systems are different (i.e in Japanese/Chinese, language involves morphemes and concepts):

"日本語が話せます"
"I can speak Japanese"

- **How can a MT model account for varying sentence length?**
  - we consider 2 (connected models):
    * an **alignment model**, which predicts which words align with which (we assume that each word in $\underline{y}$ gets aligned with exactly one word in $\underline{x}$)



    * a **translation model**, which gives translation probabilities for the aligned words. For example, if we consider bigrams we could compute:
      · $P(\text{So} \mid \text{Så})$
      · $P(\text{why} \mid \text{varför})$
      · $P(\text{are} \mid \text{våra})$

- **How can we incorporate the alignment into an n-gram model for translation?**
  - say we want to translate $\underline{x}$ into $\underline{y}$
  - we can store the alignments as a vector $\underline{a}$, such that:

$$a_i = \begin{cases} 0, & y_i \text{ doesn't align with any word in } \underline{x} \\ j, & y_i \text{ aligns with } x_j \end{cases}$$

  - then, our model involves predicting the alignment $\underline{a}$, and the translation $\underline{y}$, from $\underline{x}$; probabilistically (using the chain rule alongside an assumption of independence):

$$\begin{aligned} P(\underline{y}, \underline{a} \mid \underline{x}) &= P(\underline{y} \mid \underline{x}, \underline{a}) P(\underline{a} \mid \underline{x}) \\ &= P(|\underline{y}| \mid \underline{x}) \prod_{i=1}^{|\underline{y}|} P(y_i \mid y_{1:i-1}, \underline{x}, \underline{a}) \prod_{j=1}^{|\underline{a}|} P(a_j \mid a_{1:j-1}, \underline{x}) \end{aligned}$$

  where:

    - $P(|\underline{y}| \mid \underline{x})$ is a model for **sentence length**: it tells us the desired length of the translated sentence, given the the original sentence (notice, this doesn't have an alignment term, since $|\underline{y}| = |\underline{a}|$)
    - $\prod_{i=1}^{|\underline{y}|} P(y_i \mid y_{1:i-1}, \underline{x}, \underline{a})$ is the **translation model**
    - $\prod_{j=1}^{|\underline{a}|} P(a_j \mid a_{1:j-1}, \underline{x})$ is the **alignment model**
  - notice, since $\underline{a}$ is a **latent variable**, if we want a **conditional language model**, we need to **marginalise** the alignments:

$$P(\underline{y} \mid \underline{x}) = \sum_{\underline{a}} P(\underline{y}, \underline{a} \mid \underline{x})$$

  - if we then have a dataset $\mathcal{D}$ with N translation pairs, the **likelihood** will be:

$$\begin{aligned} P(\mathcal{D} \mid \theta) &= \prod_{n=1}^{N} P(\underline{y}^{(n)} \mid \underline{x}^{(n)}) \\ &= \prod_{n=1}^{N} \sum_{\underline{a}^{(n)}} P(|\underline{y}^{(n)}| \mid \underline{x}^{(n)}) \prod_{i=1}^{|\underline{y}|} P(y_i^{(n)} \mid y_{1:i-1}^{(n)}, \underline{x}^{(n)}, \underline{a}^{(n)}) \prod_{j=1}^{|\underline{a}^{(n)}|} P(a_j^{(n)} \mid a_{1:j-1}^{(n)}, \underline{x}^{(n)}) \end{aligned}$$

- **What is the MT workflow, according to this model?**
  - for simplicity, lets consider a **bigram** model:

$$P(|\underline{y}| \mid \underline{x}) \prod_{i=1}^{|\underline{y}|} P(a_i \mid |\underline{x}|) P(y_i \mid x_{a_i})$$

    (notice, we have combined the 2 product terms into 1)
  - for concreteness, lets consider the example above where we try to translate Swedish ($\underline{x}$) to English ($\underline{y}$):
    1. Sample possible lengths for English sentences, conditioned on the Swedish text ($P(|\underline{y}| \mid \underline{x})$)
    2. For each English word, we can draw an **alignment** with a Swedish word ($P(a_i \mid |\underline{x}|)$; typically drawing uniform samples)
    3. For each English word, sample a possible **translation** ($P(y_i \mid x_{a_i})$)
- **How is this MT scheme related to HMM?**
  - we can think of words in Swedish as a set of **states**
  - we can think of words in English as a set of **tags**
  - MT is then a HMM, where the **transition probabilities** correspond to **alignment probabilities**, and **emission probabilities** correspond to **translation probabilities**

2.2 Expectation Maximisation for Alignments

- **Why can't we directly use maximum likelihood estimation to predict the translation probabilities?**
  - for **MLE**, we need to **count** bigram occurrences
  - however, to be able to count, we need to have the **alignments**
  - $\underline{a}$ is known as a **latent variable** - we don't have its value from the data
- **What are expected counts?**

- since we don't have the alignments, we can't formally count bigram occurrences
- instead, we can use **expected counts**: for a translation pair $\underline{x}, \underline{y}$, on average, what proportion of the alignments link x_i to y_j?
- probabilitstically, this is:

$$P(a_i = j \mid \underline{x}, \underline{y}) = \frac{P(\underline{y} \mid a_i = j, \underline{x})P(a_i = j \mid \underline{x})}{P(\underline{y} \mid \underline{x})} = \frac{P(\underline{y}, a_i = j \mid \underline{x})}{P(\underline{y} \mid \underline{x})}$$

- thus, for each alignment, instead of counting 0 or 1, we use the **expected count** (which is nothing but a **posterior** probability)
- we can compute this posterior via:

$$P(a_i = j \mid \underline{x}, \underline{y}) = \frac{P(\underline{y}, a_i = j \mid \underline{x})}{P(\underline{y} \mid \underline{x})} = \frac{P(x_i \mid e_j)}{\sum_{a_i=0}^{|\underline{y}|} P(y_i \mid x_{a_i})}$$

- the **higher** the **expected count**, the more confident we are that a given alignment is good

- **Why can't we directly use expected counts when predicting the translation probabilities?**
  - to obtain **expected counts**, we need to have access to our **translation model**
  - but to get our **translation model**, we need to have access to the counts!

- **How can expectation maximisation be used to compute the parameters for our MT model?**
  - this self-referential problem calls for the use of **Expectation Maximisation**:
    1. Define some initial model θ_0
    2. Using θ_0, compute the expected counts (**expectation** step)
    3. With the expected counts, use MLE to compute the parameters of a new model, θ_1 (**maximisation** step)
    4. Continue iterating: at step i, compute θ_i by using θ_{i-1} until stopping criterion is met (i.e convergence, fixed number of iterations)
  - EM guarantees that the resulting **likelihood** will be **non-decreasing** with each new estimate for θ (theory: expectation step constructs a function which is a lower bound of the true optimal likelihood; maximisation step improves this lower bound)

- **How can the alignments be recovered from the MT model?**
  - once we have a model, finding the best alignment is relatively easy:

$$\hat{\underline{a}} = \arg\max_{\underline{a}} P(\underline{a} \mid \underline{x}, \underline{y})$$

  - componentwise, and noting that $P(a_i \mid |\underline{x}|)$ is uniform:

$$\hat{a}_i = \arg\max_{a_i} P(\underline{y} \mid \underline{x}) \prod_{i=1}^{|\underline{y}|} P(a_i \mid \underline{x}) P(y_i \mid x_{a_i}) = \arg\max_{a_i} P(y_i \mid x_{a_i})$$

2.3 Decoding with the MT Model

- **How are conditional language models trained in practice?**
  - **Bayes' Rule** is often used:

$$P(\underline{y} \mid \underline{x}) \propto P(\underline{y}) P(\underline{x} \mid \underline{y})$$

- $P(\underline{y})$ will be a **language model** (these can be trained easily)
- $P(\underline{x} \mid \underline{y})$ is our translation+alignment model (same as above, but translating in reverse)
- by training both models separately, we get the power of a good language model, alongside the translation (as opposed to just learning translation)

- **How can we decode given a conditional language model?**

1. **Greedy Search**: at step i, predict y_i to maximise:

$$P(y_i \mid y_{1:i-1}, \underline{x})$$

2. **Beam Search**: at step i, keep the k best y_i's maximising:

$$P(y_i \mid y_{1:i-1}, \underline{x})$$

- note, none of these strategies will find an **optimal** $\underline{y}$

---

*You might be wondering: given all the advances in **deep learning**, why bother on studying **n-grams**?*

1. ***Applicability**: man of these ideas will show up in NNs (maximising objective function, beam search for decoding, latent variables in unsupervised learning, alignment inspired **attention**)*
2. ***Low-Data**: when there is little data, these simpler models can perform quite well (NNs require a lot of data)*
3. ***Google**: still uses n-grams for **phrase-based translation***
4. ***Perspective**: understanding the tradeoffs of working with Markov assumptions will help you appreciate how NNs make them go away*