THE UNIVERSITY OF MELBOURNE

Comp90042 Natural Language Processing

Workshop (Week3 | 2025s1)

Dr Jean Lee

Agenda

1. Discussion (~45 mins)
   - **Text Classification**
   - **N-gram Language Model**

2. Programming (~10 mins)

3	workshop-03.pdf	03-classification.ipynb 04-ngram.ipynb

Discussion : Text classification

1. What is **text classification**? Give some examples.
   (a) Why is text classification generally a difficult problem? What are some hurdles that need to be overcome?
   (b) Consider some (supervised) text classification problem, and discuss whether the following (supervised) machine learning models would be suitable:
      i. k-Nearest Neighbour using Euclidean distance
      ii. k-Nearest Neighbour using Cosine similarity
      iii. Decision Trees using Information Gain
      iv. Naive Bayes
      v. Logistic Regression
      vi. Support Vector Machines

Text Classification

What is Text Classification?

- Is the task of classifying text documents into different labels.

Input

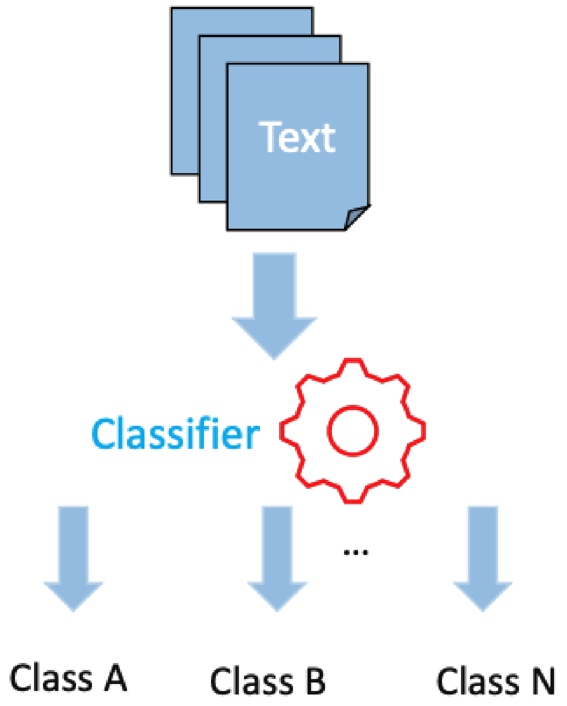
- a document d
- a fixed set of classes (labels) $C = \{c_1, c_2, \ldots, c_j\}$

(ML run) a training set of **m** labeled documents $(d_1, c_1), \ldots, (d_m, c_m)$

Output

- a predicted class $c \in C$

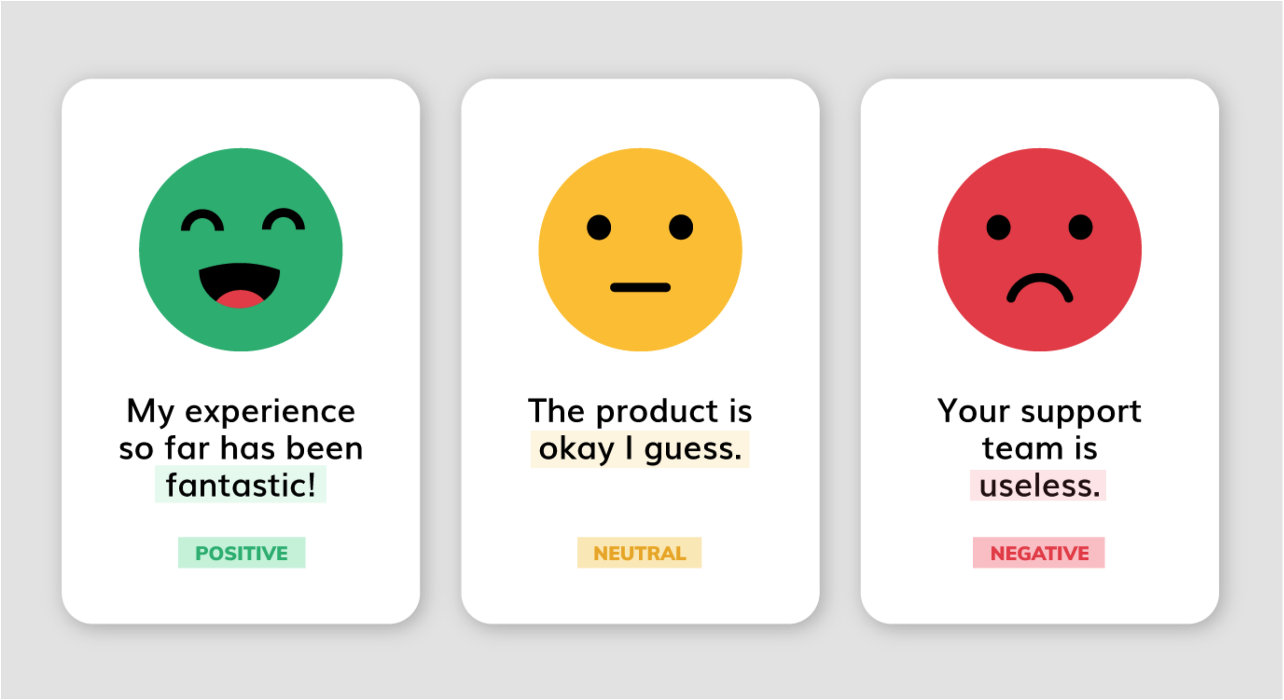
(ML run) a learned classifier $\gamma : d \rightarrow c$



Text Classification

Text Classification Examples

- Sentiment analysis
- Spam detection
- Topic classification
- Authorship identification
- Native-language identification
- Automatic fact-checking
- …



Text Classification - Challenge

Why is text classification generally a difficult problem?

- The main issue is in terms of **document representation**.
- *how do we identify **features** of the **document** which help us to distinguish between the various classes?*

- **Source** of document features: **tokens (words)** in the **document**
  - **feature selection** is often important
  - Single words: inadequate at modelling meaningful information
  - Multi-word (e.g. bi-grams): sparse data problem

Text Classification – Word Representation

How do we represent the meaning of a word?

- ***Count-based word representation :*** *(e.g.) One-hot vectors, Bag of Words, Term Frequency-Inverse Document Frequency (TF-IDF)*
- ***Prediction-based word representation :*** *(e.g.) Word2Vec, GloVe*

week5

One-hot vectors (encoding)

- regard words as discrete symbols:

  motel = [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

  hotel = [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0]

issues:

→ *no natural notion of similarity*

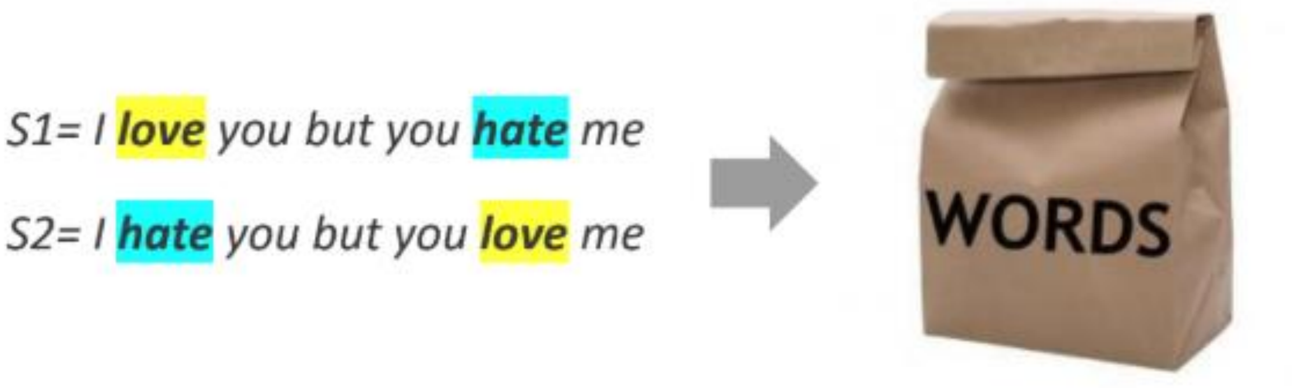
→ *number of words in vocabulary (inefficiency)*

Text Classification – Word Representation

How do we represent the meaning of a word?

Bag of Words (BoW)

- a representation of text that describes **the occurrence of words** within a document.
- The intuition is that documents are similar if they have similar content.



issues:

➔ *Discarding word order*

➔ *ignores the context*

➔ *ignores meaning of words in the document (semantics).*

Ref.: NLP lecture note (by Dr Caren Han)

Text Classification – ML models

Which ML models would be suitable for text classification?

Select one classifier and explain it to your partner. Each of you should choose a different classifier.

- k-Nearest Neighbour (kNN) using Euclidean distance
- k-Nearest Neighbour (kNN) using Cosine similarity
- Decision Trees using Information Gain
- Naive Bayes
- Logistic Regression
- Support Vector Machines

Prerequisites

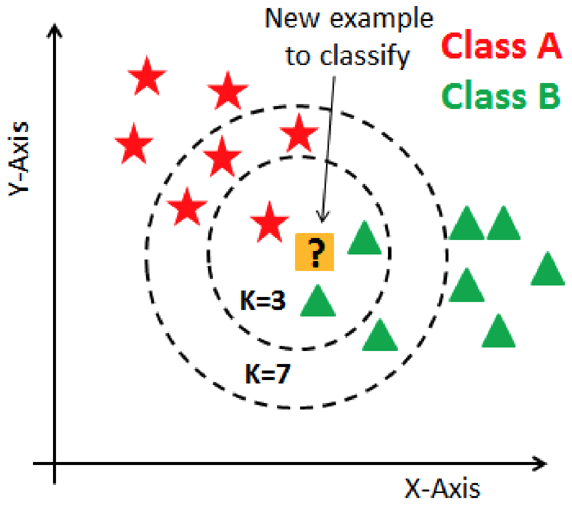
- Machine learning basics (COMP30027, COMP90049, COMP90051)
  - Modules → Welcome → Machine Learning and Linguistics Readings

https://canvas.lms.unimelb.edu.au/courses/210955/pages/machine-learning-and-linguistics-readings?module_item_id=6453877

https://www.youtube.com/watch?v=E0Hmnixke2g

k-Nearest Neighbour (kNN)

- KNN: Classify based on **majority class of k-nearest** training examples in feature space.

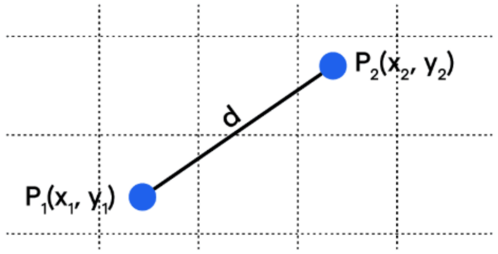


- Distance metric
- Choosing k (e.g. k =3 or ?)

(+) easy to implement, few hyperparameters

(-) not good for scale and high-dimensional data, overfitting

- k-Nearest Neighbour using **Euclidean distance**
  - Often this is a ***<u>bad idea</u>***
  - tends to classify documents **based upon their length**
  - which is usually not a distinguishing characteristic for text classification problems.

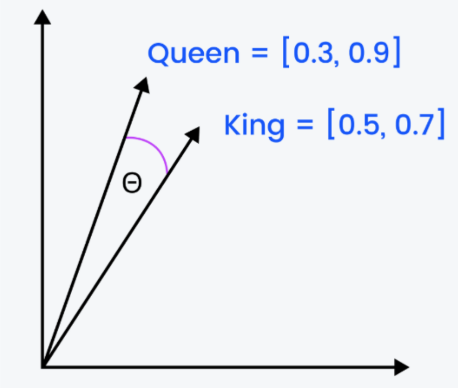


Euclidean Distance (d) $= \sqrt{(x_2 - y_1)^2 + (y_2 - y_1)^2}$

k-Nearest Neighbour (kNN)

- k-Nearest Neighbour using **Cosine similarity**
  - Usually better than using Euclidean distance
  - However, **kNN** suffers from **high-dimensionality problems**
  - our feature set based upon the presence of (all) words usually ***isn't suitable*** for this model.

- **Cosine similarity**
  - measures the similarity between two vectors of an inner (dot) product space.
  - the cosine of the angle between two vectors.



$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

$$\text{Cos(Queen, King)} = \frac{(0.3*0.5)+(0.9*0.7)}{\sqrt{0.3^2+0.9^2} * \sqrt{0.5^2+0.7^2}}$$

$$= \frac{0.15+0.63}{\sqrt{0.9^2} * \sqrt{0.74}}$$

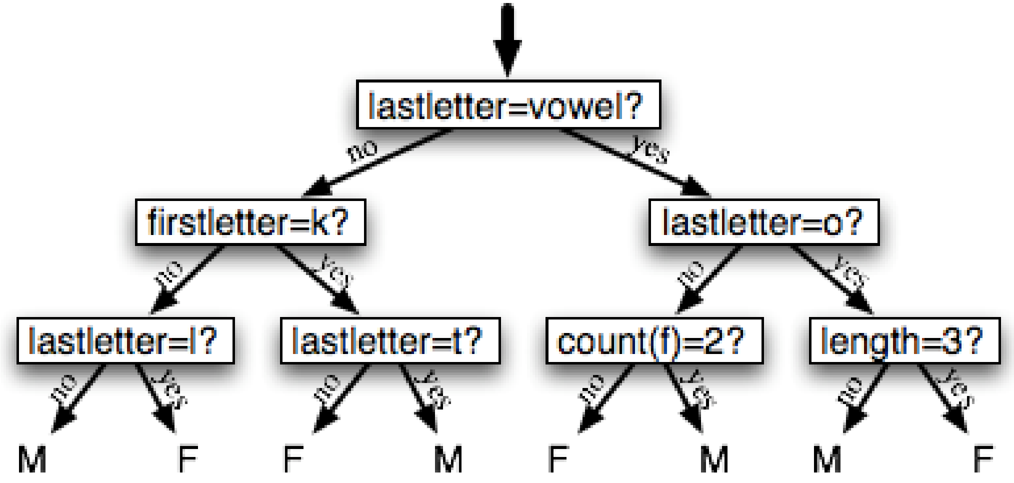
$$= \frac{0.78}{\sqrt{0.666}}$$

$$= 0.03$$

Decision Trees (DT)

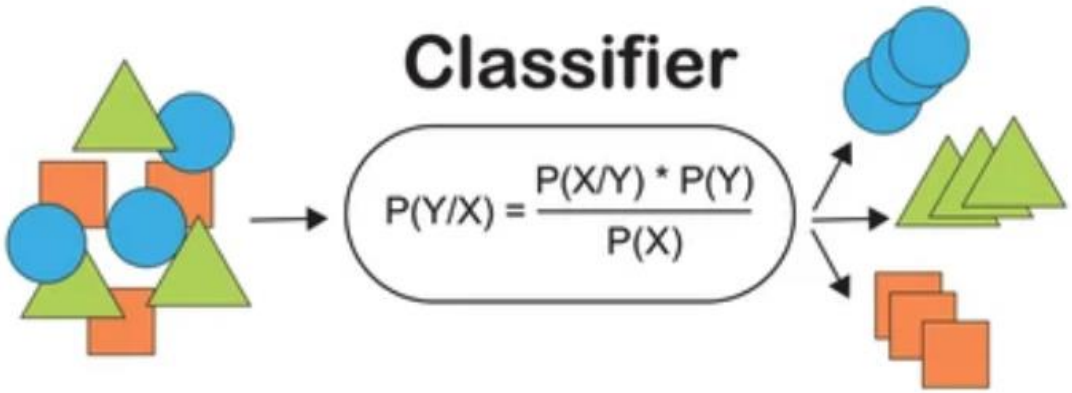
- DT : Construct a **tree where nodes correspond to tests on individual features**
  - ***can be useful*** for finding meaningful features
  - However, the feature set is very large, and we might find spurious correlations.

- Decision Trees using **Information Gain**
  - ***Information Gain :*** *to determine the best features for splitting nodes.*
  - ***poor choice*** because it tends to prefer **rare** features.



Naive Bayes (NB)

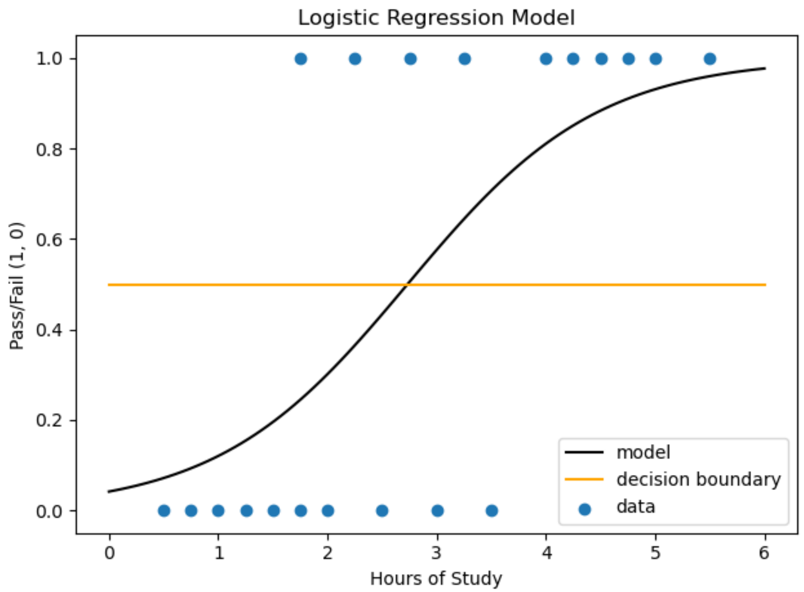
- NB : Find the class with **the highest likelihood under Bayes Law**
    - At first glance, a ***poor choice***
    - the "naive" assumption of the **conditional independence** of features and classes is highly **untrue**.
    - Also sensitive to a large feature set
    - Surprisingly somewhat ***useful anyway***!



Logistic Regression (LR)

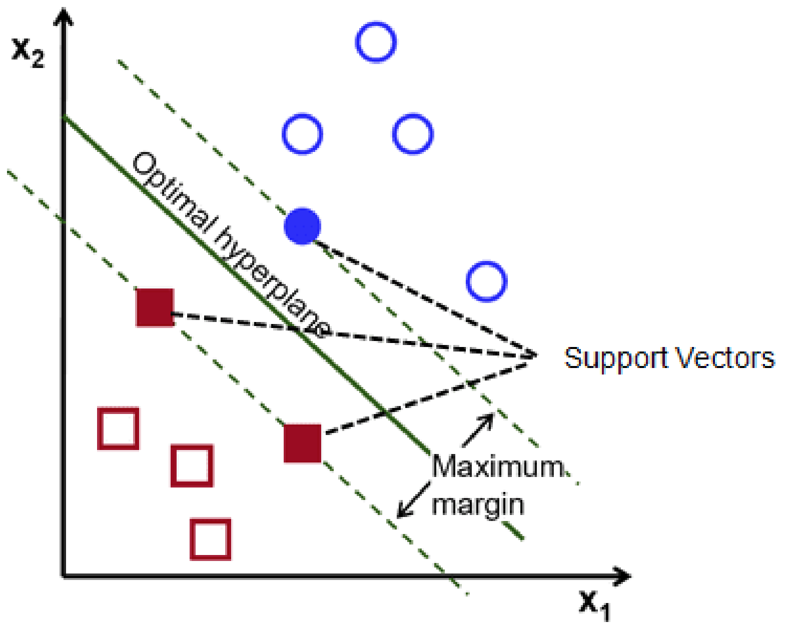
- LR : Put linear combination of features in **logistic function**
  - ***Useful***
  - it **relaxes the conditional independence requirement** of Naive Bayes.
  - Can handle large numbers of mostly useless features by **'feature weighting' step**

$$y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$



Support Vector Machines (SVM)

- SVM : Finds **hyperplane** which **separates the training data** with **maximum margin**
  - Linear kernels often quite ***effective***
  - Some combination of features are useful for characterising the classes.
  - Problems: **multiple classes** (most text classification tends to be multi–class).



Discussion : N-gram Language Model (LM)

2. For the following "corpus" of two documents:

```
1.  how much wood would a wood chuck chuck if a wood chuck
would chuck wood
2.  a wood chuck would chuck the wood he could chuck if
a wood chuck would chuck wood
```

(a) Which of the following sentences: `a wood could chuck`; `wood would a chuck`; is more probable, accoding to:

i. An unsmoothed uni-gram language model?
ii. A uni-gram language model, with Laplacian ("add-one") smoothing?
iii. An unsmoothed bi-gram language model?
iv. A bi-gram language model, with Laplacian smoothing?
v. An unsmoothed tri-gram language model?
vi. A tri-gram language model, with Laplacian smoothing?

(b) Assuming we are using a bi-gram language model with Kneser-Ney smoothing. Given the bigram `chuck a`, compute the continuation probability for `a`.

3. What does **back–off** mean, in the context of smoothing a language model? What does **interpolation** refer to?

	unsmooth	**Laplacian**
unigram	1)	2)
bi-gram	3)	4)
tri-gram	5)	6)

N-gram LM Calculation

- Corpus

1: how much wood would a wood chuck chuck if a wood chuck would chuck wood

2: a wood chuck would chuck the wood he could chuck if a wood chuck would chuck wood

- Word counts

~~<s>~~	a	chuck	could	he	how	if	much	the	wood	would	</s>	**Total**
~~2~~	4	9	1	1	1	2	1	1	8	4	2	**34**

- M: the number of all words - sum of all word frequency ***(M = 34)***
- V: the number of unique words - vocabulary size ***(V = 11)***

Why is the start symbol <s> left out?

- *used internally to set context but not included in the final n-gram probabilities*
- *because they are **artificially inserted** and **do not represent actual text content**.*
- *While the end symbol </s> is typically included because **</s> helps the model learn the likelihood of ending** after certain sequences of words.*

N-gram: 1) Unigram unsmooth

a	chuck	could	he	how	if	much	the	wood	would	</s>	**Total**
4	9	1	1	1	2	1	1	8	4	2	**34**

- M: the number of all words - sum of all word frequency ***(M = 34)***
- V: the number of unique words - vocabulary size ***(V = 11)***

- **Unigram probability (Unsmooth)**

$$P(w_i) = \frac{C(w_i)}{M}$$

Total number of **word tokens** in corpus

A: a wood could chuck

B: wood would a chuck

$$\begin{aligned} P(A) &= P(\texttt{a})P(\texttt{wood})P(\texttt{could})P(\texttt{chuck})P(\texttt{</s>}) \\ &= \frac{4}{34} \times \frac{8}{34} \times \frac{1}{34} \times \frac{9}{34} \times \frac{2}{34} \approx 1.27 \times 10^{-5} \\ P(B) &= P(\texttt{wood})P(\texttt{would})P(\texttt{a})P(\texttt{chuck})P(\texttt{</s>}) \\ &= \frac{8}{34} \times \frac{4}{34} \times \frac{4}{34} \times \frac{9}{34} \times \frac{2}{34} \approx 5.07 \times 10^{-5} \end{aligned}$$

N-gram: 2) Unigram Laplacian smoothing

a	chuck	could	he	how	if	much	the	wood	would	</s>	**Total**
4	9	1	1	1	2	1	1	8	4	2	**34**

- M: the number of all words - sum of all word frequency ***(M = 34)***
- V: the number of unique words - vocabulary size ***(V = 11)***

- **Unigram Laplacian ("add-one") smoothing**

$$P_{add1}(w_i) = \frac{C(w_i) + 1}{M + |V|}$$

A: a wood could chuck

B: wood would a chuck

$$\begin{aligned} P_L(A) &= P_L(\texttt{a})P_L(\texttt{wood})P_L(\texttt{could})P_L(\texttt{chuck})P_L(\texttt{</s>}) \\ &= \frac{5}{45} \times \frac{9}{45} \times \frac{2}{45} \times \frac{10}{45} \times \frac{3}{45} \approx 1.46 \times 10^{-5} \\ P_L(B) &= P_L(\texttt{wood})P_L(\texttt{would})P_L(\texttt{a})P_L(\texttt{chuck})P_L(\texttt{</s>}) \\ &= \frac{9}{45} \times \frac{5}{45} \times \frac{5}{45} \times \frac{10}{45} \times \frac{3}{45} \approx 3.66 \times 10^{-5} \end{aligned}$$

N-gram: 3) bi-gram unsmooth

1: how much wood would a wood chuck chuck if a wood chuck would chuck wood

2: a wood chuck would chuck the wood he could chuck if a wood chuck would chuck wood

a	chuck	could	he	how	if	much	the	wood	would	</s>	**Total**
4	9	1	1	1	2	1	1	8	4	2	**34**

- **bi-gram probability (Unsmooth)**

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

$$\begin{aligned} P(A) &= P(\text{a}|\text{<s>})P(\text{wood}|\text{a})P(\text{could}|\text{wood})P(\text{chuck}|\text{could})P(\text{</s>}|\text{chuck}) \\ &= \frac{1}{2} \times \frac{4}{4} \times \boxed{\frac{0}{8}} \times \frac{1}{1} \times \boxed{\frac{0}{9}} = 0 \\ P(B) &= P(\text{wood}|\text{<s>})P(\text{would}|\text{wood})P(\text{a}|\text{would})P(\text{chuck}|\text{a})P(\text{</s>}|\text{chuck}) \\ &= \frac{0}{2} \times \frac{1}{8} \times \frac{1}{4} \times \frac{0}{4} \times \frac{0}{9} = 0 \end{aligned}$$

A: a wood could chuck

B: wood would a chuck

N-gram: 4) bi-gram Laplacian smoothing

1: how much wood would a wood chuck chuck if a wood chuck would chuck wood

2: a wood chuck would chuck the wood he could chuck if a wood chuck would chuck wood

a	chuck	could	he	how	if	much	the	wood	would	</s>	**Total**
4	9	1	1	1	2	1	1	8	4	2	**34**

- M: the number of all words - sum of all word frequency ***(M = 34)***
- V: the number of unique words - vocabulary size ***(V = 11)***

- **bi-gram Laplacian ("add-one") smoothing**

$$P_{add1}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |V|}$$

$$P_L(A) = P_L(\text{a}|\text{<s>})P_L(\text{wood}|\text{a})P_L(\text{could}|\text{wood})P_L(\text{chuck}|\text{could}) P_L(\text{</s>}|\text{chuck}) = \frac{2}{13} \times \frac{5}{15} \times \boxed{\frac{1}{19}} \times \frac{2}{12} \times \boxed{\frac{1}{20}} \approx 2.25 \times 10^{-5}$$

$$P_L(B) = P_L(\text{wood}|\text{<s>})P_L(\text{would}|\text{wood})P_L(\text{a}|\text{would})P_L(\text{chuck}|\text{a}) P_L(\text{</s>}|\text{chuck}) = \frac{1}{13} \times \frac{2}{19} \times \frac{2}{15} \times \frac{1}{15} \times \frac{1}{20} \approx 3.60 \times 10^{-6}$$

A: a wood could chuck

B: wood would a chuck

N-gram: 5) tri-gram unsmooth

a	chuck	could	he	how	if	much	the	wood	would	</s>	**Total**
4	9	1	1	1	2	1	1	8	4	2	**34**

- M: the number of all words - sum of all word frequency ***(M = 34)***
- V: the number of unique words - vocabulary size ***(V = 11)***

- **tri-gram probability (Unsmooth)**

$$P(w_i|w_{i-1}w_{i-2}) = \frac{C(w_{i-2}w_{i-1}w_i)}{C(w_{i-2}w_{i-1})}$$

A: a wood could chuck

B: wood would a chuck

$$\begin{aligned} P(A) &= P(\texttt{a}|\texttt{<s> <s>})P(\texttt{wood}|\texttt{<s> a})\cdots P(\texttt{</s>}|\texttt{could chuck}) \\ &= \frac{1}{2}\times\frac{1}{1}\times\frac{0}{4}\times\frac{0}{0}\times\frac{0}{1} = ? \\ P(B) &= P(\texttt{wood}|\texttt{<s> <s>})P(\texttt{would}|\texttt{<s> wood})\cdots P(\texttt{</s>}|\texttt{a chuck}) \\ &= \frac{0}{2}\times\frac{0}{0}\times\frac{1}{1}\times\frac{0}{1}\times\frac{0}{0} = ? \end{aligned}$$

N-gram: 6) tri-gram Laplacian smoothing

a	chuck	could	he	how	if	much	the	wood	would	</s>	**Total**
4	9	1	1	1	2	1	1	8	4	2	**34**

- M: the number of all words - sum of all word frequency ***(M = 34)***
- V: the number of unique words - vocabulary size ***(V = 11)***

- **tri-gram Laplacian ("add-one") smoothing**

$$P_L(w_i|w_{i-1}w_{i-2}) = \frac{C(w_{i-2}w_{i-1}w_i)+1}{C(w_{i-2}w_{i-1})+V}$$

A: a wood could chuck

B: wood would a chuck

$$\begin{aligned} P_{\mathrm{L}}(A) &= P_{\mathrm{L}}(\texttt{a}|\texttt{<s> <s>})P_{\mathrm{L}}(\texttt{wood}|\texttt{<s> a})\cdots P_{\mathrm{L}}(\texttt{</s>}|\texttt{could chuck}) \\ &= \frac{2}{13}\times\frac{2}{12}\times\frac{1}{15}\times\frac{1}{11}\times\frac{1}{12} \approx 1.30\times 10^{-5} \\ P_{\mathrm{L}}(B) &= P_{\mathrm{L}}(\texttt{wood}|\texttt{<s> <s>})P_{\mathrm{L}}(\texttt{would}|\texttt{<s> wood})\cdots P_{\mathrm{L}}(\texttt{</s>}|\texttt{a chuck}) \\ &= \frac{1}{13}\times\frac{1}{11}\times\frac{2}{12}\times\frac{1}{12}\times\frac{1}{11} \approx 8.83\times 10^{-6} \end{aligned}$$

N-gram: bi-gram Kneser-Ney (KN) smoothing

- Given the bigram `chuck a`, compute the **continuation probability** for **a**

1: **how much wood would a wood chuck chuck if a wood chuck would chuck wood**

2: **a wood chuck would chuck the wood he could chuck if a wood chuck would chuck wood**

a	chuck	could	he	how	if	much	the	wood	would	</s>	**Total**
4	9	1	1	1	2	1	1	8	4	2	**34**

- **Observed bi-grams:** chuck chuck, chuck if, chuck would, chuck wood, chuck the
- **Unobserved bi-grams:** chuck a, chuck could, chuck he, chuck how, chuck much, chuck </s>

- **Continuation counts** *number of unique words in the vocabulary which appears before a word **w***
  - a = {would, if, <s>} = 3
  - could = {he} = 1
  - he = {wood} = 1
  - how = {<s>} = 1
  - much = {how} = 1
  - </s> = {wood} = 1

$$P_{cont}(\text{a}) = \frac{\#_{cont}(\text{a})}{\#_{cont}(\text{a}) + \#_{cont}(\text{could}) + \#_{cont}(\text{he}) + \#_{cont}(\text{how}) + \#_{cont}(\text{much}) + \#_{cont}(\text{</s>})}$$

$$= \frac{3}{3+1+1+1+1+1}$$

***Numerator:** continuation count of (w_i)*

***Denominator:** Sum of the continuation count of all unobserved word for (w_{i-1})*

N-gram: bi-gram Kneser-Ney (KN) smoothing

- bi-gram probability (Unsmooth)

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i)}{C(w_{i-1})}$$

- bi-gram Laplacian ("add-one") smoothing

$$P_{add1}(w_i \mid w_{i-1}) = \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |V|}$$

- bi-gram Kneser-Ney (KN) smoothing

$$P_{KN}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}, w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}, w_i) > 0 \\ \beta(w_{i-1}) \boxed{P_{cont}(w_i),} & \text{otherwise} \end{cases}$$

the amount of probability mass that has been discounted for context w_{i-1}

$$P_{cont}(w_i) = \frac{|\{w'_{i-1} : C(w'_{i-1}, w_i) > 0\}|}{\sum_{\{w_j : C(w_{i-1}, w_j) = 0\}} |\{w_{j-1} : C(w_{j-1}, w_j) > 0\}|}$$

N-gram: back-off and interpolation

Back-off

- Use **lower-order n-gram model** if higher-order is unseen.
- For example, if we have never seen some tri-gram from our sentence, we can instead consider the bigram probability.

interpolation

- Take **weighted average sum of n-gram**.
- Instead of only “falling back” to lower (back-off), consider every probability as a linear combination of all of the relevant n-gram models.

$$
\begin{aligned}
\mathrm{p}_{\text{Interpolation}}(w_m \mid w_{m-1}, w_{m-2}) = {} & \lambda_3 \mathrm{p}_3^*(w_m \mid w_{m-1}, w_{m-2}) \\
& + \lambda_2 \mathrm{p}_2^*(w_m \mid w_{m-1}) \\
& + \lambda_1 \mathrm{p}_1^*(w_m).
\end{aligned}
$$

Programming!

Programming

1. In the `03-classification` notebook, observe how different tokenisation regimes alter the text classification performance of the various classifiers on the given Reuters dataset problem.

2. Using the iPython notebook `04-ngram`, randomly generate some sentences based on the bi-gram models of the Gutenberg corpus and the Penn Treebank. What do you notice about these sentences? Are there any sentences which might get returned for both corpora? Why?