This lecture

• An extractive summary of the course.

• Open office hours will follow, 12:30-1:30 zoom (Zining) and in-person (Professor Lee)
Exam

• Saturday April 22, 2023 from 14h00—17h00.
• **No aids allowed** – your desk should have nothing but:
  • Your UofT ID,
  • The exam, and
  • A writing implement.
Structure

• Following the format of previous years:
  • 20*2 **multiple-choice** questions [40 marks]
    • 4 options each.
  • 10*3 **short-answer** questions [30 marks]
    • Some of these involve simply giving a definition. Others involve some calculation.
  • 3*10 **subject-specific** questions [30 marks]
    • These questions involve a small component of original thinking.
8. Melamed’s method of sentence alignment works by ...
   (a) minimizing the costs of alignments according to the lengths of the aligned sentences.
   (b) minimizing the costs of alignments according to the lengths of the aligned words.
   (c) estimating cognates based on 4-graphs.
   (d) estimating cognates based on longest common subsequences.

9. Greedy decoding in statistical machine translation iteratively updates the best guess of the English translation $E^*$, given the French sentence $F$, according to ...
   (a) transformations of words and alignments.
   (b) transformations of words only.
   (c) the total cost of alignment.
   (d) the total number of matching cognates.

10. Which of these phonemes is not voiced?
    (a) /b/.
    (b) /ih/.
    (c) /m/.
    (d) /k/.

11. The Nyquist rate is ...
    (a) the rate at which the glottis vibrates.
    (b) twice the rate at which the glottis vibrates.
    (c) twice the maximum frequency preserved in a sampled signal.
    (d) twice the sampling rate of a sampled signal.

12. Which feature is known to correlate positively with a sentence’s selection into an extractive text summary in the news domain?
    (a) Early position in the document being summarized.
    (b) High function-word to content-word ratio.
    (c) High number of stigma words.
    (d) None of the above.
Short answer

2. State Bayes’s Rule.

3. Name and define the three types of text-to-speech synthesis architectures. Give one advantage each architecture has over the others.
We can work it out

SMT 2. (5 marks)

Given the two reference translations below, compute the BLEU score for each of the two candidate translations, assuming that you only consider unigrams and bigrams, and that there is no cap. *Hint:* Your results should be of the form $x^y$ where $x$ is a fraction or some other term, and $y$ is a positive or negative fraction.

**Reference 1** Use the Force Luke

**Reference 2** Use some Force Luke

**Candidate 1** Use some of the Force

**Candidate 2** Use the Force
Hints for studying

• Definitions: *n.pl.* Terms that are useful to know.
  
  • Highlights are also useful to know.

• Not all definitions/highlights are in the exam.
• Not all things on the exam have been highlighted.
  • This review lecture is likewise not a substitute for the rest of the material in this course.
Hints for studying

• Go through the lectures from this year.

• Work out worked-out examples for yourself, ideally more than once.

• I find it helpful to relax before an exam.
Exam material

• The exam covers all material in the *lectures and assignments except:
  • Material in the bonuses of assignments, and
  • Slides with ‘Aside’ in the title.

• The reading material (e.g., Manning & Schütze) provides background to concepts discussed in class.
  • If a concept appears in a linked paper but not in the lectures/assignments, you don’t need to know it, even if it’s very interesting.
# Exam Topics

<table>
<thead>
<tr>
<th>Introduction to corpus-based linguisticsf</th>
<th>hapax legomenon</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-gram, Linguistic Features, Classification</td>
<td>MLE, add-delta smoothing, perplexity, tokenization</td>
</tr>
<tr>
<td>Entropy and Information Fheory</td>
<td>entropy, information gain, mutual information, t-test, Bonferroni</td>
</tr>
<tr>
<td>Neural language models and word embedding</td>
<td>neural network, CBO, ELM, gender definitional</td>
</tr>
<tr>
<td>Machine translation (statistical and neural) (MT)</td>
<td>cognates</td>
</tr>
<tr>
<td>Recent Breakthroughs - (SOTA) transformer variants.</td>
<td></td>
</tr>
<tr>
<td>HMMs</td>
<td>Baum-Welch, greedy decoding</td>
</tr>
<tr>
<td>Automatic speech recognition (ASR)</td>
<td>Noisy Channel Model, Levenshtein distance</td>
</tr>
<tr>
<td>Natural Language Understanding (NLU)</td>
<td>Discourse, Dataset</td>
</tr>
<tr>
<td>Information retrieval (IR)</td>
<td>document frequency, doc2vec, evaluation score</td>
</tr>
<tr>
<td>Interpretability and LLM</td>
<td>Shapley value, generate explanations, improving LLM</td>
</tr>
</tbody>
</table>
2018 Final exam distribution
Part 1. Text Classification
Categories of linguistic knowledge

• **Phonology:** the study of patterns of speech sounds.
  e.g., “read” → /r iy d/

• **Morphology:** how words can be changed by inflection or derivation.
  e.g., “read”, “reads”, “reader”, “reading”, ...

• **Syntax:** the ordering and structure between words and phrases.
  e.g., *NounPhrase* → *det. adj. n.*

• **Semantics:** the study of how meaning is created by words and phrases.
  e.g., “book”

• **Pragmatics:** the study of meaning in broad contexts.
Corpora

• **Corpus**: *n.* A body of language data of a particular sort (*pl.* corpora).

• Most **valuable** corpora occur **naturally**
  - e.g., newspaper articles, telephone conversations, multilingual transcripts of the United Nations

• We use corpora to gather statistics; more is better (typically between $10^7$ and $10^{12}$ tokens).
Notable corpora

• **Penn treebank**: Syntactically annotated Brown, plus others incl. 1989 *Wall Street Journal*.
• **Switchboard corpus**: 120 hours ≈ 2.4M tokens. 2.4K telephone conversations between US English speakers.
• **Hansard corpus**: Canadian parliamentary proceedings, French/English bilingual.
Very simple predictions

• A model at the heart of SMT, ASR, and IR...
• We want to know the probability of the next word given the previous words in a sequence.

• We can approximate conditional probabilities by counting occurrences in large corpora of data.
  • E.g., \( P(\text{food} \mid I \text{ want Chinese}) = \frac{P(I \text{ want Chinese food})}{P(I \text{ want Chinese)}} \approx \frac{\text{Count}(I \text{ want Chinese food})}{\text{Count}(I \text{ want Chinese})} \)
Bayes’ theorem

\[ P(A, B) = P(A)P(B|A) \]

\[ P(A, B) = P(B)P(A|B) \]

Bayes theorem: \( P(A|B) = \frac{P(B|A)P(A)}{P(B)} \)
Maximum likelihood estimate

- Maximum likelihood estimate (MLE) of parameters \( \theta \) in a model \( M \), given training data \( T \) is the estimate that maximizes the likelihood of the training data using the model.

- e.g., \( T \) is the Brown corpus, \( M \) is the bigram and unigram tables, \( \theta_{(to|want)} \) is \( P(to|want) \).
Sparsity of unigrams vs. bigrams

- E.g., we’ve seen lots of every unigram, but are missing many bigrams:

<table>
<thead>
<tr>
<th>Unigram counts:</th>
<th>Count($w_{t-1},w_t$)</th>
<th>$w_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count($w_{t-1},w_t$)</td>
<td>I</td>
<td>want</td>
</tr>
<tr>
<td>I</td>
<td>2533</td>
<td>927</td>
</tr>
<tr>
<td>Count($w_{t-1},w_t$)</td>
<td>Unigram</td>
<td>Hapax legomena: n.pl. words that occur once in a corpus.</td>
</tr>
<tr>
<td>I</td>
<td>want</td>
<td>to</td>
</tr>
<tr>
<td>I</td>
<td>5</td>
<td>827</td>
</tr>
<tr>
<td>want</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>to</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>eat</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Chinese</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>food</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>lunch</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>spend</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Zipf’s law on the Brown corpus

\[ f \propto \frac{1}{r} \]

i.e., for some \( k \)

\[ f \cdot r = k \]

From Manning & Schütze
Smoothing as redistribution

- Steal from the rich and give to the poor.
- E.g., $\text{Count}(I\ caught\ \cdot)$
Add-1 smoothing (Laplace)

- Given a vocab size $||\mathcal{V}||$ and corpus size $N$, just add 1 to all the counts! No more zeros!

- MLE: $P(w_i) = \frac{C(w_i)}{N}$

- Laplace estimate: $P_{Lap}(w_i) = \frac{C(w_i)+1}{N+||\mathcal{V}||}$

- Does this give a proper probability distribution? Yes:

$$\sum_w P_{Lap}(w) = \sum_w \frac{C(w) + 1}{N + ||\mathcal{V}||} = \frac{\sum_w C(w) + \sum_w 1}{N + ||\mathcal{V}||} = \frac{N + ||\mathcal{V}||}{N + ||\mathcal{V}||} = 1$$
Add-$\delta$ smoothing

- Laplace’s method generalizes to the add-$\delta$ estimate:

$$P_\delta(w_i) = \frac{C(w_i) + \delta}{N + \delta \|V\|}$$

- Consider also:
  - Simple interpolation
  - Katz smoothing
  - Good-Turing smoothing
Extrinsic evaluation

• The **utility** of a **language model** is often determined *in situ* (i.e., in **practice**).
  
  • e.g.,
    1. **Alternately embed** LMs $A$ and $B$ into a **speech recognizer**.
    2. **Run** speech recognition using each model.
    3. **Compare** recognition rates between the system that uses LM $A$ and the system that uses LM $B$.
Intrinsic evaluation

• To measure the **intrinsic value** of a language model, we first need to estimate the **probability of a corpus**, $P(C)$.
  • This will also let us **adjust/estimate** model **parameters** (e.g., $P(to|want)$) to maximize $P(Corpus)$.

• For a **corpus** of sentences, $C$, we sometimes make the assumption that the **sentences are conditionally independent**: $P(C) = \prod_i P(s_i)$
Intrinsic evaluation

- We estimate $P(\cdot)$ given a particular corpus, e.g., Brown.
- A good model of the Brown corpus is one that makes Brown very likely (even if that model is bad for other corpora).

\[
P_1(to|want) = \ldots
\]

If
\[
P_1(\text{Brown corpus}) \geq P_j(\text{Brown corpus}) \quad \forall j
\]

then

$P_1$ is the best model of the Brown corpus.
Perplexity

- Perplexity corp. $C$, $PP(C) = 2^{-\frac{\log_2 P(C)}{||C||}} = P(C)^{-1/||C||}$

- If you have a vocabulary $\mathcal{V}$ with $||\mathcal{V}||$ word types, and your LM is uniform (i.e., $P(w) = \frac{1}{||\mathcal{V}||}$ $\forall w \in \mathcal{V}$),

- Then

$$PP(C) = 2^{-\frac{\log_2 P(C)}{||C||}} = 2^{-\frac{\log_2 \left(\frac{1}{||\mathcal{V}||}\right)}{||C||}} = 2^{-\log_2 \left(\frac{1}{||\mathcal{V}||}\right)} = 2^{\log_2 ||\mathcal{V}||} = ||\mathcal{V}||$$

- Perplexity is sort of like a ‘branching factor’.

- Minimizing perplexity $\equiv$ maximizing probability of corpus
Perplexity as an evaluation metric

• **Lower** perplexity → a **better** model.
  • (more on this in the section on information theory)

• e.g., splitting WSJ corpus into a 38M word training set and a 1.5M word test set gives:

<table>
<thead>
<tr>
<th>N-gram order</th>
<th>Unigram</th>
<th>Bigram</th>
<th>Trigram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perplexity</td>
<td>962</td>
<td>170</td>
<td>109</td>
</tr>
</tbody>
</table>
Different features for different tasks

- **Alzheimer’s disease** involves atrophy in the brain.
  - Excessive **pauses** (acoustic disfluencies),
  - Excessive **word type repetition**, and
  - Simplistic or **short** sentences.
    - ‘**function words**’ like *the* and *an* are often **dropped**.
- **To diagnose** Alzheimer’s disease, one might measure:
  - **Proportion** of utterance spent in **silence**.
  - **Entropy** of **word type** usage.
  - **Number of word tokens** in a sentence.
    - **Number of prepositions** and **determiners** (explained shortly).

**Explainability/Interpretability!**
Features in Sentiment Analysis

- **Sentiment analysis** can involve detecting:
  - **Stress** or **frustration** in a conversation.
  - **Interest**, **confusion**, or **preferences**. Useful to marketers.
    - e.g., ‘*got socks for xmas wanted #ps5 fml*’
  - **Decept**. e.g., ‘*Let’s watch Netflix and chill.*’

- Complicating factors include **sarcasm**, **implicitness**, and a **subtle** spectrum from **negative** to **positive** opinions.

- **Useful features** for sentiment analyzers include:
  - Trigrams.
  - First-person pronouns.
Information and Entropy
Entropy

- **Entropy**: \( n. \) the **average** amount of information we get in observing the output of source \( S \).

\[
H(S) = \sum_i p_i I(w_i) = \sum_i p_i \log_2 \frac{1}{p_i}
\]

Note that this is **very** similar to how we define the expected value (\( \text{i.e., } \) ‘average’) of something:

\[
E[X] = \sum_{x \in X} p(x) x
\]
Joint entropy

- **Joint Entropy:** *n.* the **average** amount of information needed to specify multiple variables simultaneously.

\[
H(X, Y) = \sum_x \sum_y p(x, y) \log_2 \frac{1}{p(x, y)}
\]

Same general form as entropy, except you sum over each variable, and probabilities are joint.
Conditional entropy

• **Conditional entropy**: \( n. \) the average amount of information needed to specify one variable \textbf{given} that you know another.

\[
H(Y|X) = \sum_{x \in X} p(x)H(Y|X = x)
\]

It’s an average of entropies over all possible conditioning values.
Relations between entropies

\[ H(X,Y) = H(X) + H(Y) - I(X;Y) \]
Mutual information

**Mutual information:** \( n. \) the **average** amount of information shared between variables.

\[
I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)
\]

\[
= \sum_{x,y} p(x, y) \log_2 \frac{p(x,y)}{p(x)p(y)}
\]

Again, a sum over each variable, but the log fraction is normalized by an assumption that they’re independent \((p(x)p(y))\).
Information theory

- In general, lectures includes some walked-through examples of applying the preceding formula.
- It’s probably a good idea to walk through these examples yourself on paper.
Procedure of a statistical test

Step 1: **State** a hypothesis.
- Null hypothesis $H_0$ and alternative hypothesis $H_1$; more in the next slide.

Step 2: **Compute** some test statistics.
- For example: p-value

Step 3: **Compare** the statistics to a critical value and report the test results.
- E.g., compare p to $\alpha = 0.05$ (“**significance level**”). If $p<0.05$, reject $H_0$. Otherwise, do not reject $H_0$. 
Null and Alternative Hypotheses

- **Null hypothesis** $H_0$ usually states that “nothing has changed”.

- **Alternative hypothesis** $H_1$ usually states that “there are some meaningful findings”.

Summary: Types of $t$-tests

- **One-sample $t$-test**: whether the population mean equals $\mu$.
  - Population mean $X$ is a random variable.
  - `scipy.stats.ttest_1samp`

- **Two-sample $t$-test**: whether the mean of two populations, $X$ and $Y$, equal each other.
  - `scipy.stats.ttest_ind`

- **Paired $t$-test**: whether $X - Y$ equals a known value $\mu$.
  - `scipy.stats.ttest_rel`
Observable Markov model
Multivariate systems

• What if a conditioning variable changes over time?
  • e.g., I’m **happy** one second and **disgusted** the next.
• Here, the **state** is the mood and the **observation** is the word.

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>upside</td>
<td>0.1</td>
</tr>
<tr>
<td>down</td>
<td>0.05</td>
</tr>
<tr>
<td>promise</td>
<td>0.05</td>
</tr>
<tr>
<td>friend</td>
<td>0.6</td>
</tr>
<tr>
<td>monster</td>
<td>0.05</td>
</tr>
<tr>
<td>midnight</td>
<td>0.1</td>
</tr>
<tr>
<td>hallow</td>
<td>0.05</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>word</th>
<th>P(word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>upside</td>
<td>0.3</td>
</tr>
<tr>
<td>down</td>
<td>0</td>
</tr>
<tr>
<td>promise</td>
<td>0</td>
</tr>
<tr>
<td>friend</td>
<td>0.2</td>
</tr>
<tr>
<td>monster</td>
<td>0.05</td>
</tr>
<tr>
<td>midnight</td>
<td>0.05</td>
</tr>
<tr>
<td>hallow</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Observable multivariate systems

Q: How do you learn these probabilities?

\[ P(w_{0:t}, q_{0:t}) \approx \prod_{i=0}^{t} P(q_i|q_{i-1}) P(w_i|q_i) \]

A: Basically, the same as before.

\[ P(q_i|q_{i-1}) = \frac{P(q_{i-1}q_i)}{P(q_{i-1})} \] is learned with MLE from training data.

\[ P(w_i|q_i) = \frac{P(w_i,q_i)}{P(q_i)} \] is also learned with MLE from training data.
Hidden variables

- Q: What if you **don’t** have access to the **state** during testing?
  - e.g., you’re asked to compute $P(\langle up, up \rangle)$

- Q: What if you **don’t** have access to the **state** during **training**?
Tasks for HMMs

1. Given a model with particular parameters $\theta = \langle \Pi, A, B \rangle$, how do we efficiently compute the likelihood of a particular observation sequence, $P(\mathcal{O}; \theta)$?

2. Given an observation sequence $\mathcal{O}$ and a model $\theta$, how do we choose a state sequence $Q = \{q_0, \ldots, q_T\}$ that best explains the observations?

3. Given a large observation sequence $\mathcal{O}$, how do we choose the best parameters $\theta = \langle \Pi, A, B \rangle$ that explain the data $\mathcal{O}$?
1. Trellis

The diagram represents a trellis structure with states $s_1$, $s_2$, $s_3$, ..., $s_N$. The states are indexed over time $t = 0, 1, 2, ..., T-1$. The probability of being in state $s_3$ at time $t = 2$ is highlighted.
I want to guess which sequence of states generated an observation. 

E.g., if states are PoS and observations are words.
2. The Viterbi algorithm

• Also an inductive dynamic-programming algorithm that uses the trellis.

• Define the probability of the most probable path leading to the trellis node at \((\text{state } i, \text{ time } t)\) as

\[
\delta_i(t) = \max_{q_0 \ldots q_{t-1}} P(q_0 \ldots q_{t-1}, \sigma_0 \ldots \sigma_{t-1}, q_t = s_i; \theta)
\]

• And the incoming arc that led to this most probable path is defined as \(\psi_i(t)\)
3. Training HMMs

- We want to modify the parameters of our model \( \theta = (\Pi, A, B) \) so that \( P(O; \theta) \) is maximized for some training data \( O \):

\[
\hat{\theta} = \arg\max_{\theta} P(O; \theta)
\]

- If we want to choose a best state sequence \( Q^* \) on previously unseen test data, the parameters of the HMM should first be tuned to similar training data.
Expecting and maximizing

If we knew $\theta$, we could make **expectations** such as
- Expected number of times in state $s_i$,
- Expected number of transitions $s_i \rightarrow s_j$

If we knew:
- Expected number of times in state $s_i$,
- Expected number of transitions $s_i \rightarrow s_j$

then we could compute the **maximum likelihood estimate** of

$$\theta = \langle \pi_i, \{a_{ij}\}, \{b_i(w)\} \rangle$$
Baum-Welch re-estimation

- **Baum-Welch (BW):** a specific version of **EM** for HMMs. a.k.a. ‘**forward-backward**’ algorithm.

1. Initialize the model.
2. **E-step:** Compute **expectations** for $Count(q_{t-1}q_t)$ and $Count(q_t \land w_t)$ given model, training data $O$.
3. **M-step:** Adjust our **start**, **transition**, and **observation** probabilities to **maximize** the likelihood of $O$.
4. **Go to 2.** and repeat until convergence or stopping condition...
Statistical Machine Translation

STICK ONE IN YOUR EAR, YOU CAN INSTANTLY UNDERSTAND ANYTHING SAID TO YOU IN ANY FORM OF LANGUAGE: THE SPEECH YOU HEAR DECODES THE BRAIN WAVE MATRIX.
Challenges of SMT

- Lexical ambiguity (e.g., words are polysemous).
- Differing word orders.
- Syntactic ambiguity.
- Miscellaneous idiosyncrasies.
The noisy channel

Language model

Source $P(E)$

Translation model

Channel $P(F|E)$

Decoder

Observed $F$

$E^* = \arg\max_E P(F|E)P(E)$
Sentence alignment by cognates

• Cognates: *n.pl.* Words that have a common etymological origin.

• Etymological: *adj.* Pertaining to the historical derivation of a word. E.g., *porc* → *pork*

• The intuition is that words that are related across languages have similar spellings.
  • e.g., *zombie/zombie, government/gouvernement*
  • Not always: *son* (male offspring) vs. *son* (sound)

• Cognates can “anchor” sentence alignments between related languages.
Greedy Decoding

- Core idea: take the most probable word on each step

$$y_t = \arg\max_i (p_{t,i})$$

- Problem: Can’t recover from a prior bad choice (no ‘undo’)

- Sub-optimal in an auto-regressive setup:
  - $\tilde{h}_t$ continuous, depends on $y_{t-1}$
  - DP (optimal sequence) solutions for discrete, finite state spaces (e.g. Viterbi search - HMM lecture) impossible
Bilingual evaluation: BLEU

- In lecture, $||\text{Ref1}|| = 16$, $||\text{Ref2}|| = 17$, $||\text{Ref3}|| = 16$, and $||\text{Cn1}|| = 18$ and $||\text{Cn2}|| = 14$,

\[
brevity_1 = \frac{17}{18} \quad BP_1 = 1
\]

\[
brevity_2 = \frac{16}{14} \quad BP_2 = e^{1 - \left(\frac{8}{7}\right)} = 0.8669
\]

- Final score of candidate $C$:

\[
\text{BLEU} = BP \times (p_1 p_2 \ldots p_n)^{1/n}
\]

where

\[
p_n = \frac{\sum_{\text{ngram} \in C} \text{Count}_R(\text{ngram})}{\sum_{\text{ngram} \in C} \text{Count}_C(\text{ngram})}
\]
BLEU example

• **Reference 1:**  
  I am afraid Dave

• **Reference 2:**  
  I am scared Dave

• **Reference 3:**  
  I have fear David

• **Candidate:**  
  I fear David

Assume $\text{cap}(n) = 2$ for all $n$-grams

- brevity $= \frac{4}{3} \geq 1$ so $BP = e^{1 - \left(\frac{4}{3}\right)}$

- $p_1 = \frac{\sum_{\text{1-gram} \in C} \text{Count}_R(1\text{gram})}{\sum_{\text{1-gram} \in C} \text{Count}_C(1\text{gram})} = \frac{1+1+1}{1+1+1} = 1$

- $p_2 = \frac{\sum_{\text{2-gram} \in C} \text{Count}_R(2\text{gram})}{\sum_{\text{2-gram} \in C} \text{Count}_C(2\text{gram})} = \frac{1}{2}$

- $\text{BLEU} = BP(p_1p_2)^{\frac{1}{2}} = e^{1 - \left(\frac{4}{3}\right)} \left(\frac{1}{2}\right)^{\frac{1}{2}} \approx 0.5067$
Beam search: top-$K$ greedy

- **Core idea**: track the $K$ top choices (most probable) of partial translations (or, hypotheses) at **each step** of decoding

- $K$ is also called the ‘beam width’ or ‘beam size’
  - Where, $5 \leq K \leq 10$ usually in practice

- The score of a hypothesis $(y_1, ..., y_t)$ is its log probability:

$$score(y_1, ..., y_t) = \log P_{LM}(y_1, ..., y_t|x) = \sum_{i=1}^{t} \log P_{LM}(y_i|y_1, ..., y_{i-1}, x)$$

- We search and track the top $k$ hypotheses based on the **score**
- Scores are all negative, and higher is better

- Beam search is **not guaranteed** to find the optimal solution
- However, much more **efficient and practical** than exhaustive search
Neural Language Models
"You shall know a word by the company it keeps."
— J.R. Firth (1957)

\[ P(w_t = \text{lugubrious} | w_{t-1} = \text{feeling}, w_{t-2} = \text{been, ...}) \]

been feeling lugubrious all day
felt a lugubrious sadness in
...

Here, we’re predicting the center word given the context. This is called the ‘continuous bag of words’ (CBOW) model\(^1\).

https://code.google.com/p/word2vec/
**Continuous bag of words (1 word context)**

Note: we have two vector representations of each word:

\[ \mathbf{v}_w = \mathbf{x}^T \mathbf{W}_I \] (\(w^{th}\) row of \(\mathbf{W}_I\))

\[ \mathbf{v}_w = \mathbf{W}_O^T \mathbf{y} \] (\(w^{th}\) col of \(\mathbf{W}_O\))
Continuous bag of words ($C$ words context)

- If we want to use more context, $C$, we need to change the network architecture somewhat.
  - Each input word will produce one of $C$ embeddings.
  - We just need to add an intermediate layer, usually this just averages the embeddings.
Importance of in-domain data

Let’s talk about gender at the UofT

However, in word2vec trained on Google News, 
man:woman::programmer:homemaker.

Recurrent neural networks

• Consider RNNs generally, and LSTMs and others, specifically
• **Hint**: How do these models differ and how they are similar? What are their **strengths** and **weaknesses**?
• What are the components of an LSTM network?
Long short-term memory (LSTM)

- Within each recurrent unit or cell:
  - Self-looping recurrence for **cell state using vector C**
  - Information flow regulating structures called **gates**

\[
\begin{align*}
    f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
    i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
    \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
    C_t &= f_t \ast C_{t-1} + i_t \ast \tilde{C}_t
\end{align*}
\]
ELMo: Embeddings from Language Models

• Instead of a fixed embedding for each word type, ELMo considers the entire sentence before embedding each token.
• It uses a bi-directional LSTM trained on a specific task.
• Outputs are softmax probabilities on words, as before.
ELMo: Embeddings from Language Models

- Instead of a fixed embedding for each word type, ELMo considers the entire sentence before embedding each token.
- It uses a bi-directional LSTM trained on a specific task.
- Outputs are softmax probabilities on words, as before.

Automatic Speech Recognition
Levenshtein distance

<table>
<thead>
<tr>
<th>Reference</th>
<th>-</th>
<th>how</th>
<th>to</th>
<th>wreck</th>
<th>a</th>
<th>nice</th>
<th>beach</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>how</td>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>to</td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>recognize</td>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>speech</td>
<td></td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

• See the example in lecture. **Work it out yourself.**
NLU, IR, Interpretablility

CSC401/2511 – Natural Language Computing – Winter 2023
University of Toronto
“Hierarchies” of understanding

Humans can understand languages on multiple “levels”.

● These levels are not mutually exclusive!

● Many NLP tasks require understanding at multiple levels (e.g., translation).

● These levels are here for organizing the tasks.
Discourse structure in Abstract

NLP tasks, such as question-answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset – matching or exceeding the performance of 3 out of 4 baseline systems without using the 170,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from the naturally occurring demonstrations.
Similarity score

• If the query and the available documents can be represented by vectors, we can determine similarity according to their cosine distance.
  • Vectors that are near each other (within a certain angular radius) are considered relevant.
Vectorization: \textit{tf.idf}

- \textit{tf.idf} is a traditional method to vectorize the documents.
- It starts by weighting \textit{words} in the \textit{documents}.

- **Term frequency, \( tf_{ij} \):** number of occurrences of word \( w_i \) in document \( d_j \).

- **Document frequency, \( df_i \):** number of documents in which \( w_i \) appears.

- **Collection frequency, \( cf_i \):** total occurrences of \( w_i \) in the collection.
Term frequency

• Higher values of $tf_{ij}$ (for contentful words) suggest that word $w_i$ is a good indicator of the content of document $d_j$.
  • When considering the relevance of a document $d_j$ to a keyword $w_i$, $tf_{ij}$ should be maximized.

• We often dampen $tf_{ij}$ to temper these comparisons.
  • $tf_{dampen} = 1 + \log(tf)$, if $tf > 0$. 
Document frequency

- The **document frequency**, \( d_f_i \), is the number of documents in which \( w_i \) appears.
  - **Meaningful** words may occur repeatedly in a related document, but **functional** (or less meaningful) words may be distributed evenly over all documents.

<table>
<thead>
<tr>
<th>Word</th>
<th>Collection frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel</td>
<td>10,440</td>
<td>3997</td>
</tr>
<tr>
<td>try</td>
<td>10,422</td>
<td>8760</td>
</tr>
</tbody>
</table>

- E.g., *kernel* occurs about as often as *try* in total, but it occurs in fewer documents – it is a more **specific** concept.
Inverse document frequency

• Very specific words, \( w_i \), would give smaller values of \( df_i \).

• To maximize specificity, the inverse document frequency is

\[
idf_i = \log \left( \frac{D}{df_i} \right)
\]

where \( D \) is the total number of documents
and we scale with log (why? next slide)

• This measure gives full weight to words that occur in 1 document, and zero weight to words that occur in all documents.
Inverse document frequency

• The probability of a document containing word $i$ is:

$$\frac{df_i}{D}$$

“A document containing word $i$” is an event.
Small $p$: this event is more surprising.
Therefore, more information

• $idf_i$ is the amount of information provided by observing the event.
tf.idf vectorization of a document

- We combine the term frequency and the inverse document frequency to give us a joint measure of relatedness between words and documents:

\[
    tf.idf(w_i, d_j) = \begin{cases} 
        (1 + \log(tf_{ij})) \log \frac{D}{df_i} & \text{if } tf_{ij} \geq 1 \\
        0 & \text{if } tf_{ij} = 0
    \end{cases}
\]

- The \(j^{th}\) document is therefore represented by a vector:

\[
    [tf.idf(w_1, d_j), \quad tf.idf(w_2, d_j), \quad \ldots, \quad tf.idf(w_{|W|}, d_j)]
\]
Evaluating the retrieval systems

- Some commonly used metrics include:
  - Precision
  - Recall
  - F-score
  - Precision @ k
Interpretable NLP

CSC401/2511 – Natural Language Computing – Winter 2023
Lecture 12
University of Toronto
Shapley value $\phi_k$

- Imagine $n$ players are playing a game with $\gamma$ as the result.

  - The Shapley value $\phi_k$ is the contribution / importance of $x_k$:
    - How much $x_k$ can change $\gamma$.

Lloyd Shapley won Nobel Memorial Prize in Economics in 2012
Shapley value $\phi_k$

- Consider a set ("coalition") of players that do not contain $x_k$:
  
  $$S \subseteq \{x_1 \ldots x_n\} \setminus x_k$$

- How would the outcome $y$ differ if these players play with $x_k$?
  
  $$y_{\cup x_k} - y_S$$

- The Shapley value $\phi_k$ is the expectation of such difference:
  
  $$\phi_k = \mathbb{E}[y_{\cup x_k} - y_S]$$
Shapley value $\phi_k$

\[ \phi_k = \mathbb{E}[y_{\cup x_k} - y_S] = \sum_{s \in \{x_1 \ldots x_n\} \setminus x_k} p(S)[y_{\cup x_k} - y_S] \]

where:

\[ p(S) = \frac{|S|! (n - 1 - |S|)!}{n!} \]
Question: An elephant can’t be put into a fridge because it is too large. What is it?

(A) elephant  
(B) fridge

Answer: (A) elephant

My explanation: An elephant is too large to be put into a fridge, so the pronoun “it” refers to the subject, “elephant”.

Explanation provides common-sense knowledge.
NLE as Seq2Seq generation

Here is an example and an expected explanation.

An elephant can’t be put into a fridge because it is too large. It refers to the elephant because

Needs some prompt engineering here
Final thoughts

(not thoughts on the final)
Natural Language Processing in Industry
Final thoughts

• This course **barely** scratches the surface of these beautiful topics. Talk to these people:

  Graeme Hirst  
  Gerald Penn  
  Suzanne Stevenson  
  Yang Xu  
  Annie Lee

• Many of the techniques in this course are applicable **generally**.
• Now is a great time to make fundamental **progress** in this and adjacent areas of research.
Aside – Knowledge

• **Anecdotes** are often useless except as proofs by contradiction.
  • E.g., “I saw Google used as a verb” does **not** mean that Google is **always** (or even **likely** to be) a verb, just that it is **not always** a noun.

• **Shallow statistics** are often not enough to be truly meaningful.
  • E.g., “My ASR system is 95% accurate on my test data. Yours is only 94.5% accurate, you horrible knuckle-dragging idiot.”
    • What if the test data was **biased** to favor my system?
    • What if we only used a **very small** amount of data?

• We need a **test** to see if our statistics actually **mean** something.

Find some way to be **comfortable** making **mistakes**
Thank you