AdvNLP/E Seminar 2

Language Modelling 1

Dr Julie Weeds, Spring 2026

Sentence Completion Challenge (SCC)

- We will be playing a version of the SCC online
- In each question:
  - Fill in the gap in the sentence with the word which seems most likely.
  - All of the sentences are taken from Sherlock Holmes novels.
  - You will be given a choice of 5 words (and 10s to choose which you think is the most likely word in the context).

Ready to start

- Go to **https://PollEv.com/julieweeds376**

Sentence Completion Challenge

- Discuss the provided Sentence Completion Challenge questions with your group – what makes some of them more difficult:-
  - for you?
  - for a computer?

	Sentence	a	b	c	d	e
1	A few faint _____ were gleaming in a violet sky.	tragedies	stars	rumours	noises	explanations
2	I tell you that he is a _____ and dangerous man.	venerable	delicate	glorious	clever	sweet
3	The butler brought me my _____ into the library, and I took the chance to ask him a few questions.	entrance	affairs	coffee	pickle	nose
4	He used to make _____ over the cleverness of women, but I have not heard him do it of late.	progress	matters	merry	advances	rules
5	He had heard nothing, and the ____ remained a complete mystery.	affair	devil	snow	challenge	illusion
6	Then he _____ over the hill.	vanished	bent	leant	wept	hovered
7	I am sorry to have _____ you.	killed	convinced	practised	expected	interrupted
8	The coachman saw him _____ the hall.	neglect	strike	cross	adjust	grasp
9	Presently he emerged, looking even more _____ than before	numerous	unprofitable	instructive	flurried	reassuring
10	The door at the bottom was _____, but unlocked.	closed	destroyed	iron	insidious	empty
11	Holmes' voice _____ as he spoke	stared	sank	hummed	limped	shot
12	Holmes rushed into the crowd to _____ the lady; but just as he reached her he gave a cry and dropped to the ground, with the blood freely running down his face.	protect	conquer	replace	misunderstand	abandon
13	A maid _____ across and threw open the window.	dwelt	talked	rushed	slept	fainted
14	The King _____ at him in amazement.	rejoiced	knocked	stared	smiled	landed
15	I knew that it was my _____ voice.	unreasonable	mightier	sister's	weak	gambling

Group discussion– 5 mins

Language models

PREVIOUSLY

- Lexical and Distributional semantics
  - semantic relationships
  - WordNet
  - distributional hypothesis

THIS TIME

- Probabilistic language models
  - n-gram modelling
  - evaluation and perplexity
  - generation
  - generalization and smoothing

Lecture 2.1 Questions

1. Give 3 examples of applications where one might want to assign a probability to a sequence of words.
2. Write down the product of probabilities which would need to be calculated to estimate the probability of "*Then he hovered over the hill*" using a trigram model.
3. Is measuring perplexity for a language model an example of intrinsic or extrinsic evaluation?
4. The perplexity of model A is measured as 85 and the perplexity of model B is measured as 143. Which is better and why?

Group discussion: 5-10 mins

2.1.1 Why do we want to be able to assign a probability to a sentence?

- Machine translation

  P(high winds tonight) > P(large winds tonight)

- Spelling correction

  P(The office is about 15 **minutes** from my house)
  > P(The office is about 15 **minuets** from my house)

- Speech recognition

  P(I saw a van) > P(eyes awe of an)

2.1.2 Unigram model

- n = 1

$$P(w_1, w_2, w_3, \dots, w_k) = \prod_{i=1}^{k} P(w_i)$$

$$P(\text{“Then he hovered over the hill”}) = P(\text{“Then”}) \times P(\text{“he”}) \times P(\text{“hovered”}) \times P(\text{“over”}) \times P(\text{“the”}) \times P(\text{“hill”})$$

2.1.2 Bigram model

- n=2

$$P(w_1, w_2, w_3, \dots, w_k) = \prod_{i=1}^{k} P(w_i | w_{i-1})$$

$$P(\text{“Then he hovered over the hill”})$$
$$= P(\text{“Then”}|START) \times P(\text{“he”}|\text{“Then”}) \times P(\text{“hovered”}|\text{“he”})$$
$$\times P(\text{“over”}|\text{“hovered”}) \times P(\text{“the”}|\text{“over”}) \times P(\text{“hill”}|\text{“the”})$$

2.1.2 Trigram model

- n=3

$$P(w_1, w_2, w_3, \dots, w_k) = \prod_{i=1}^{k} P(w_i | w_{i-2}, w_{i-1})$$

$$P(\text{“Then he hovered over the hill”})$$
$$= P(\text{“Then”} | START, START) \times P(\text{“he”} | START, \text{“Then”})$$
$$\times P(\text{“hovered”} | \text{“Then”}, \text{“he”}) \times P(\text{“over”} | \text{“he”}, \text{“hovered”})$$
$$\times P(\text{“the”} | \text{“hovered”}, \text{“over”}) \times P(\text{“hill”} | \text{“over”}, \text{“the”})$$

2.1.3 Extrinsic vs Intrinsic evaluation

- EXTRINSIC
  - Put each model in a task which requires a language model
    - spelling correction
    - machine translation
    - speech recognition
  - Run the task and get an accuracy for each model
    - how many misspelt words corrected properly?
    - how many words translated correctly?
- INTRINSIC
  - Evaluate each model according to how well it models language
  - Does the model assign higher probabilities to seen sentences than to unseen sentences?

2.1.3 Perplexity

- The best language model is one that best predicts an unseen test set
  - returns the highest P(sentences)
- Perplexity is the inverse probability of the test set, normalised by the number of words

$$PP(W) = P(w_1, w_2, w_3, \ldots, w_N)^{-1/N}$$

$$PP(W) = e^{-1/N \log P(w_1, w_2, w_3, \ldots, w_N)}$$

- this assumes that we have calculated probability as a sum of logs
- multiplying by -1/N first and then raising e to this power, makes the computation possible with floating point numbers

2.1.4 Minimising perplexity

- Example:
  - training 38 million words, testing 1.5 million words (WSJ text)

	unigram	bigram	trigram
Perplexity	962	170	109

Maximising probability is the same as minimising perplexity

- Perplexity should only really be compared for the same training and testing corpora

Lecture 2.2 Questions

1. Explain how a bigram language model can be used to generate possible sequences of tokens.
2. Why do language models need to be smoothed?
3. What does OOV stand for? How do we smooth a language model with respect to OOV tokens?
4. Name 2 different methods for smoothing the probabilities of combinations of tokens. Explain one of them.

Group discussion: 5-10 mins

2.2.1 Generation

- The Shannon-Visualisation Method
  - Choose a random bigram (_ST.,w) according to its probability
  - Now choose another random bigram (w,x) according to its probability
  - And so on until we choose _END
  - Then string the words together

_ST.	I					
	I	want				
		want	to			
			to	eat		
				eat	Chinese	
					Chinese	food

I want to eat Chinese food

2.2.2 Zeros (sparsity)

TRAINING SET

- … denied the allegations
- … denied the reports
- … denied the claims
- … denied the request

TEST SET

- … denied the loan
- … denied the offer

$$P(\text{“offer”} \mid \text{“denied the”}) = 0$$

Trigrams (or even bigrams or unigrams) with zero probability in the training set mean that we

- assign zero probability to test set
- cannot calculate perplexity

2.2.3 Unknown words

- Test corpus contains words that the training corpus doesn't
- Training corpus also contains words that the test corpus doesn't
- Which training corpus words are least likely to be in the test corpus?
- Fix the vocabulary (top *N* words in training corpus or all words which occur *f* or more times)
- Create a **<UNK>** token which captures probabilities for ***Out-Of-Vocabulary (OOV)*** words.

2.2.4 Absolute discounting interpolation

- If we subtract *d* from each bigram count, how much probability mass do we save for unobserved bigrams?
- We need to keep track of the discounts made for each word
  - each time we discount a bigram ***c(w₂|w₁)***, we add that discount to a dummy token ***lambda*** for that word $\boldsymbol{c(\lambda|w_1)}$
  - normalise counts as probability distributions as before
  - For a smoothed probability estimate of any bigram, **interpolate**

    sum the observed (discounted) probability and a proportion of reserved probability mass (according to the unigram probability of w_2)

$$P_e(w_2|w_1) = P_d(w_2|w_1) + P_d(\lambda|w_1) \times P(w_2)$$

2.2.4 Stupid backoff

- Can apply absolute discounting interpolation to web-scale language models
- But Brants et al. (2007) showed that a much simpler algorithm might be sufficient at this scale
- Stupid backoff gives up on the idea of making it a true probability distribution
- No discounting of higher order probabilities
- If a higher-order n-gram has a zero count, simply "*backoff*" to a lower order n-gram, with a fixed weight ($\lambda = 0.4$)

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if count}(w_{i-k+1}^{i}) > 0 \\ \lambda S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

More questions?

Discussion of paper

- The Microsoft Research Sentence Completion Challenge (Zweig and Burges, 2011)

Zweig and Burges (2011) introduce an evaluation task known as the Microsoft Research Sentence Completion Challenge. We will be looking at this task and its potential to evaluate language models. There are 1040 *questions* in the dataset, an example of which is given below:

> **Was she his [client ‖ musings ‖ discomfiture ‖ choice ‖ opportunity], his friend or his mistress?**

1. Explain what the task is, for a human or a computer system, for a question, as presented above. In the above example, what knowledge is needed in order to choose the correct answer?

2. How was the dataset created? How and why were the incorrect answers selected in the way they were?

3. How is performance measured on this task? What score must a method achieve to be better than the *baseline* of random guessing?

4. What are the advantages and disadvantages of this evaluation task compare to ones based on human synonymy judgements such as WordSim353?

5. How does the simple 4-gram model work?

6. What do you understand by a *smoothed* n-gram model?

7. Explain the method based on latent semantic analysis similarity. Why do you think this does better than the n-gram methods?

8. How do you think you could do better on this task (without asking humans to help!)?

References

Geoffrey Zweig and Christopher Burges. 2011. The microsoft research sentence completion challenge. Technical report, Microsoft Research, December.

Group discussions: 15-30 mins

Question 1

- ***"Was she his [client || musings || discomfiture || choice || opportunity], his friend or mistress?"***

- With reference to this example, explain what the Sentence Completion Task is. In this example, what knowledge is needed in order to choose the correct answer?

Interest in semantic modeling for text is growing rapidly (see for example [1, 2, 3, 4]). However, currently there are few publicly available large datasets with which researchers can compare results, and those that are available focus on isolated word pairs. For example, WordSimilarity-353 [5] consists of 353 word pairs whose degree of similarity has been determined by human judges. In [6], the authors make available a test set consisting of 950 questions in which the goal is to find the word that is most opposite in meaning to another.

As a step towards addressing this problem, we present a set of 1,040 English sentences, taken from five novels written by Sir Arthur Conan Doyle. Each sentence has associated with it four *impostor* sentences, in which a single (fixed) word in the original sentence has been replaced by an impostor word with similar occurrence statistics. For each sentence the task is then to determine which of the five choices for that word is the correct one. The task is thus similar to a language SAT test. Our dataset was constructed from 19th century novel data from Project Gutenberg. We chose to use this source because of the high quality of the English, and also to avoid any copyright issues. We chose to use a single author (Conan Doyle) for the target sentences to give a consistent style of writing. We plan to construct similar datasets in the future to help explore other axes (multiple authors, and modern English, such as is typical in Wikipedia). Our data can be found at

Question generation was done in two steps. First, a candidate sentence containing an infrequent word was selected, and alternates for that word were automatically determined by sampling with an n-gram language model. The n-gram model used the immediate history as context, thus resulting in words that make "look good" locally, but for which there is no a-priori reason to expect them to make sense globally. In the second step, we eliminated choices which are obviously incorrect because they constitute grammatical errors. Choices requiring semantic knowledge and logical inference were preferred, as described in the guidelines, which we give in section 3. Note that an important *desideratum* guiding the data generation process was requiring that a researcher who knows exactly how the data was created, including knowing which data was used to train the language model, should nevertheless not be able to use that information to solve the problem. We now describe the data that was used, and then describe the two steps in more detail.

Question 2

- How was the dataset created?
- How and why were the incorrect answers selected in the way they were?

Step 1: Select seed sentences

Seed sentences were selected from five of Conan Doyle's Sherlock Holmes novels: *The Sign of the Four (1890), The Hound of the Baskervilles (1892), The Adventures of Sherlock Holmes (1892), The Memoirs of Sherlock Holmes (1894),* and *The Valley of Fear (1915).* Once a focus word within the sentence was selected, alternates to that word were generated using a n-gram language model. This model was trained on approximately 540 texts from the Project Gutenberg collection, consisting mainly of 19th century novels. Of these 522 had adequate headers attesting to lack of copyright, and they are now available the *Sentence Completion Challenge* website.

Step 2: Automatically generate alternatives

Alternates were generated for every sentence containing an infrequent word. A state-of-the-art class-based maximum entropy n-gram model [7] was used to generate the alternates. The following procedure was used:

1. Select a word with overall frequency less than 10^{-4}. For example, we might select "extraordinary" in "It is really the most extraordinary and inexplicable business."
2. Use the two-word history immediately preceding the selected focus word to predict alternates. We sampled 150 unique alternates at this stage, requiring that they all have frequency less than 10^{-4}. For example, "the most" predicts "handsome" and "luminous."
3. If the original (correct) sentence has a better score than any of these alternates, reject the sentence.
4. Else, score each option according to how well it and its immediate predecessor predict the next word. For example, the probability of "and" following "most handsome" might be 0.012.
5. Sort the predicted words according to this score, and retain the top 30 options.

Step 3: Human grooming (see paper for complete list of instructions

The human judges (who picked the best four choices of impostor sentences from the automatically generated list of thirty) were given the following instructions:

1. All chosen sentences should be grammatically correct. For example: *He dances while he ate his pipe* would be illegal.
2. Each correct answer should be unambiguous. In other words, the correct answer should always be a significantly better fit for that sentence than each of the four impostors; it should be possible to write down an explanation as to why the correct answer is the correct answer, that would persuade most reasonable people.
3. Sentences that might cause offense or controversy should be avoided.

Question 3

- How is performance measured on this task?
- What score must a method achieve to be better than the baseline of random guessing?

Method	**% Correct** (N=1040)
Human	91
Generating Model	31
Smoothed 3-gram	36
Smoothed 4-gram	39
Simple 4-gram	34
Average LSA Similarity	49

Table 1 Summary of Benchmarks

As a step towards addressing this problem, we present a set of 1,040 English sentences, taken from five novels written by Sir Arthur Conan Doyle. Each sentence has associated with it four *impostor* sentences, in which a single (fixed) word in the original sentence has been replaced by an impostor word with similar occurrence statistics. For each sentence the task is then to determine which of the five choices for that word is the correct one. The task is thus similar to a language SAT test. Our dataset was constructed from 19th century novel data from Project Gutenberg. We chose to use this source because of the high quality of the English, and also to avoid any copyright issues. We chose to use a single author (Conan Doyle) for the target sentences to give a consistent style of writing. We plan to construct similar datasets in the future to help explore other axes (multiple authors, and modern English, such as is typical in Wikipedia). Our data can be found at *http://research.microsoft.com/scc/*.

Question 4

- What are the advantages and disadvantages of this evaluation task compared to ones based on human synonymy judgements such as WordSim353?

Compared to human synonymy judgements

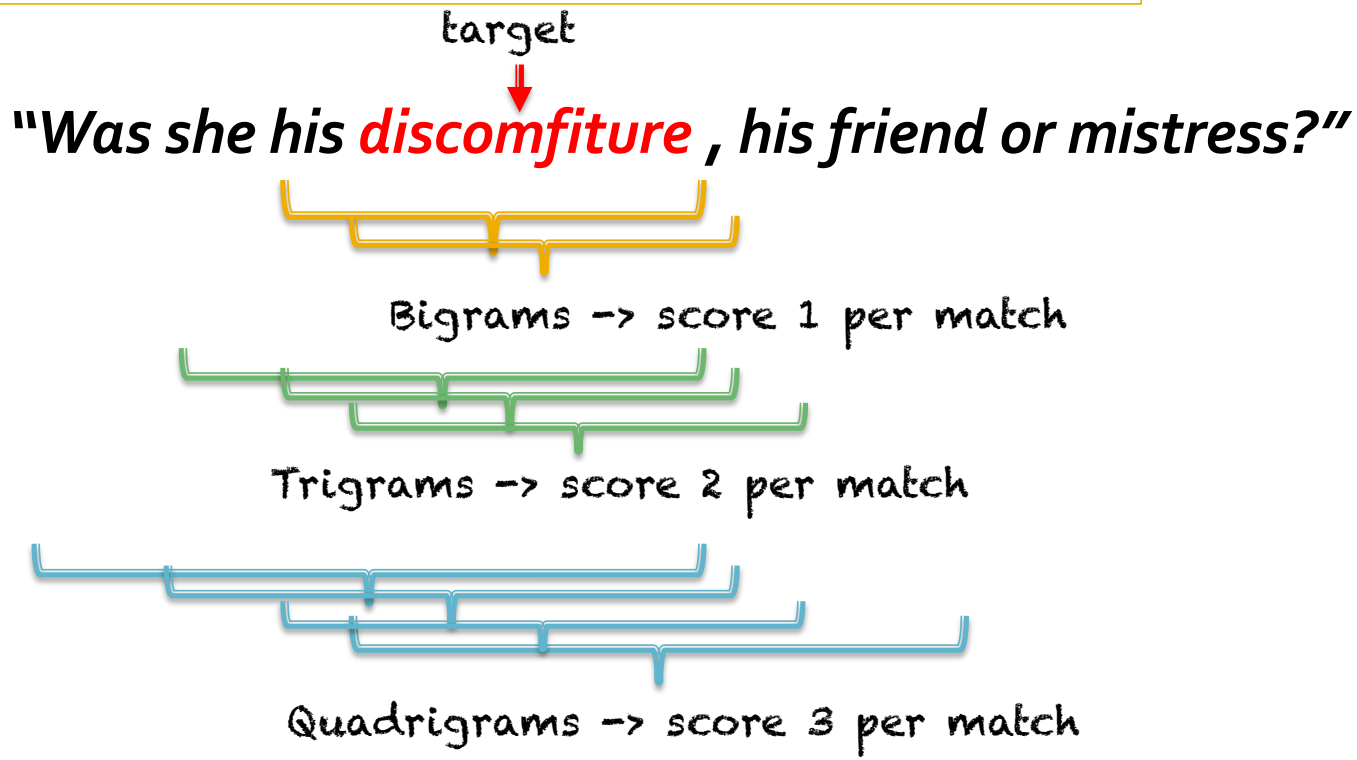
- Advantages
- Disadvantages

Question 5

- How does the simple 4-gram model work?

4.1 A Simple 4-gram model

As a sanity check we constructed a very simple N-gram model as follows: given a test sentence (with the position of the target word known), the score for that sentence was initialized to zero, and then incremented by one for each bigram match, by two for each trigram match, and by three for each 4-gram match, where a match means that the N-gram in the test sentence containing the target word occurs at least once in the background data. This simple method achieved 34% correct (compared to 20% by random choice) on the test set.



Question 6

- What do you understand by a *smoothed* n-gram model?

4.2 Smoothed N-gram model

As a somewhat more sophisticated baseline, we use the CMU language modeling toolkit [1] to build a 4-gram language model using Good-Turing smoothing. We kept all bigrams and trigrams occurring in the data, as well as four-grams occurring at least twice. We used a vocabulary of the 126k words that occurred five or more times, and this resulted in a total of 26M N-grams. This improved by 5% absolute on the simple baseline to achieve 39% correct.

4-gram language model

$$P(w_1, w_2, w_3, \ldots, w_k) = \prod_{i=1}^{k} P(w_i | w_{i-3}, w_{i-2}, w_{i-1})$$

Good-Turing -> another discounting technique
- Rather than a fixed discount, re-estimate frequency of occurrence of an n-gram which occurs r times as:

$$r* = (r+1)\frac{n_{r+1}}{n_r}$$

where n_r is the number of N-grams occurring r times

Question 7

- Explain the method based on latent semantic analysis similarity. Why do you think this does better than the n-gram methods?

4.3 Latent Semantic Analysis Similarity

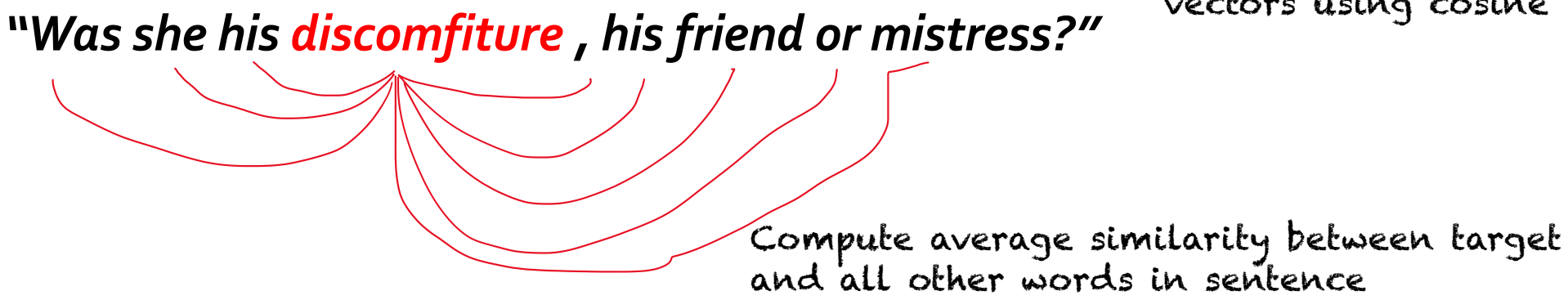
As a final benchmark, we present scores for a novel method based on latent semantic analysis. In this approach, we treated each sentence in the training data as a "document" and performed latent semantic analysis [8] to obtain a 300 dimensional vector representation of each word in the vocabulary. Denoting two words by their vectors $\mathbf{x}, \mathbf{y}$, their similarity is defined as the cosine of the angle between them:

$$\mathtt{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\| \mathbf{x} \| \| \mathbf{y} \|}.$$

To decide which option to select, we computed the average similarity to every other word in the sentence, and then output the word with the greatest overall similarity. This results in our best baseline performance, at 49% correct.

LSA ->

- Build a co-occurrence vector for each word
- This is the same as in distributional semantics but here the "co-occurrence" is a document or a sentence
- High dimensionality - number of sentences in the corpus
- Use dimensionality reduction technique (more in week 4) to reduce to 300 dimensions
- Compute similarity between vectors using cosine



Question 8

- How do you think you could do better on this task (without asking humans to help!)?

Coming up

- Neural language models (week 3)
  - feed-forward
  - RNNs and LSTMs
  - character-based

- More distributional semantics (week 4)
  - Dimensionality reduction
  - Word embeddings (word2vec, GloVe)

References

- Brants, T. et al. 2007. Large language models in machine translation. *In EMNLP/CONLL 2007*
- Church, K.W. and Gale, W.A. 1991. A Comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5, 19-54
- Mikolov, Yih and Zweig 2013 – Linguistic Regularities in Continuous Space Word Representations, (NAACL-HCT 2013)
- Zweig, G. and Burges, A. 2011. The Microsoft Research Sentence Completion Challenge. Microsoft Technical Report