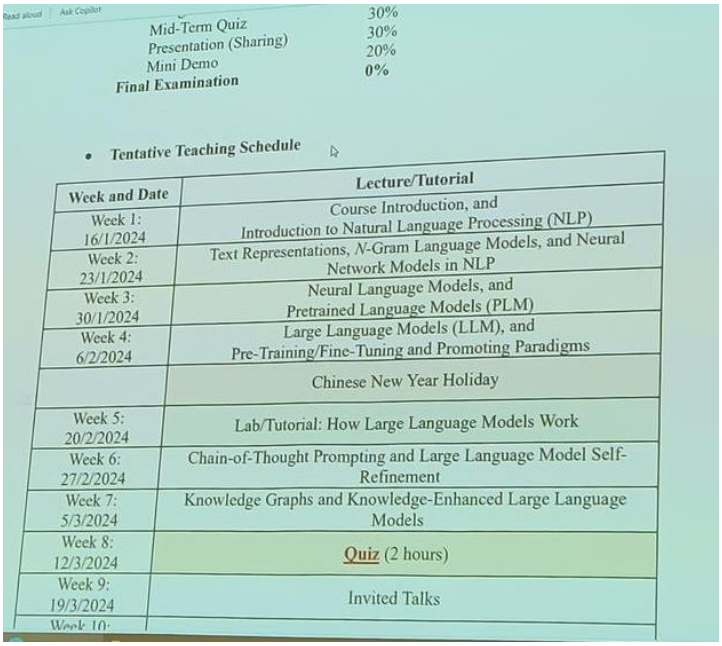

Mid-Term Quiz	30%
Presentation (Sharing)	30%
Mini Demo	20%
Final Examination	**0%**

- **Tentative Teaching Schedule**

Week and Date	Lecture/Tutorial
Week 1: 16/1/2024	Course Introduction, and Introduction to Natural Language Processing (NLP)
Week 2: 23/1/2024	Text Representations, *N*-Gram Language Models, and Neural Network Models in NLP
Week 3: 30/1/2024	Neural Language Models, and Pretrained Language Models (PLM)
Week 4: 6/2/2024	Large Language Models (LLM), and Pre-Training/Fine-Tuning and Promoting Paradigms
	Chinese New Year Holiday
Week 5: 20/2/2024	Lab/Tutorial: How Large Language Models Work
Week 6: 27/2/2024	Chain-of-Thought Prompting and Large Language Model Self-Refinement
Week 7: 5/3/2024	Knowledge Graphs and Knowledge-Enhanced Large Language Models
Week 8: 12/3/2024	Quiz (2 hours)
Week 9: 19/3/2024	Invited Talks
Week 10:	

Lec1

---

2 top conferences in computational linguistics: ACL, EMNLP (Empirical Methods 实证方法 in NLP)

NLP: Natural Language (Human Language) + Processing (Application), CS ∩ Linguistic => NLP

Info Extraction: 信息提取, Info retrieval: 信息检索

NLP hierarchy: morphological analysis (语素), syntactic analysis (语法), Semantic analysis (语义), Discourse analysis

Morphological analysis: Tokenization / Word segmentation, Lemmatization/Stemming (词形还原/词根)

Syntactic analysis: part-of-speech tagging (PoS 词性), syntactic parsing (assign syntactic structure)

Semantic analysis: (sentence level) Word sense disambiguation (WSD), Semantic role tagging (assign roles to spans in sentences), Abstract Meaning Representation (AMR),

Discourse analysis: co-reference resolution (He -> ?), discourse coherence (语篇连贯性)

Tasks: NL Understanding, NL Generation, Textual Entailment (Classification)(TE) & Natural Language Inferencing (NLI), Info extraction / summarization (eg, event extraction), Knowledge Graphs (a network of real-world entities)

Technologies: Text Classification (Sentiment / Emotion Extraction, Spam/Fake Detection), Text ranking (for Info retrieval, extractive summary, retrieval based dialog system), Info extract and Seq labelling (for QA, Knowledge discovery)

Lec2

History

Symbolic NLP 1950~1990 grammar models, logical form (syntactics), semantics, (Linguistic) rule-based

-> Statistical NLP 1990~2010 corpus, ML based, feature engineering

-> Neural NLP (2010s ~ present), NN, DL, Repr learning (self-supervised)

ELIZA (1964, first chatbot, pattern matching, transformation rules, totally rule based)

Challenges

Document Representation

Text normalization / preprocessing (morphological, syntactic analysis) -> Text representation (e.g., feature vector):

1.Sentence segmentation (most useful: punctuations (?.!), !: ambiguous for period(.), can be rule based)

2.Word Tokenization (word types & tokens (i.e., occurrences), word segmentation (e.g., Chinese, may contain overlapping (90%, *计算 机会* 下象棋)/grouping (10%, *学生 会* 听 老师 的) ambiguity), Sub-word based tokenization (token smaller than words, to deal with sparse word, ambig word, e.g, byte pair encoding BPE))

3.word normalization: words/tokens -> standard formation, e.g., (acronym:) colour->color, Cat->cat, (time normalization:) 3 Dec 2014-> 03/12/2014

4.word lemmatization/stemming: morphological analysis -> stems & affixes, reduce vocab inflectional/derivational morphology

Lemmatization: translating -> translat*e*, stemming: translating -> translat (good for info retrieval, easy impl)

5.bag of word (BOW) representation, incl representative words (mainly n., v.) & non~ (no concrete meaning, “stop word” in info retrieval, may to be removed)

Binary weight scheme (Vector Space Models)

TF-IDF weight scheme: $W_{t,d} = tf_{t,d} * idf_t$

Term freq ($tf_{t,d}$) of term t: number of occurrence in document d

Document freq df_t: number of documents containing a t.

A word occur in less number of documents -> a more important indicative word

⇨ Inverse Document Frequency (IDF): $idf_t = \log(10; N/df_t)$

(Okapi) BM25 Weighting Scheme: weighted tf + IDF

Word representation: not suitable for one-hot (long vector, sparse representation, meaning zero similarity), e.g., Synonymy (love/like, couch/sofa), similar meaning (coffee/tea, cat/dog), relation (coffee/cup)

Distributed representations: co-occurrence matrix (how many times 2 words co-occur, symmetric) -> Semantic Similarity: cosine (a, b) = a·b / (|a||b|) (Still long vector, sparse representation)

⇨ Embedding (e.g., LSA) -> dense representation

Word2Vec, e.g, CBOW, Skip-Gram, both make use of contextual information

Target word with neighboring context -> + examples, with other random words -> - examples.

Logistic regression to train a classifier to distinguish -> weights as embeddings (self-supervised)

P(+| w, c) + P(-| w, c) = 1

Similarity (w, c) = t·c --> sigmoid --> P(+ | w, c)

Skip-gram model: learn 2 separate embedding for each word w: target embedding t (W), context embedding c (C). Tries to shift embeddings so the embeddings are closer.

Static embedding: learn one fixed embedding, e.g., static.

Contexture embedding: different embedding for different context. E.g., ELMo, BERT. Can use: conv, seq, Fully-connected graph-based model

N-Gram language model (statistical model)

Language model: Next word prediction, word sequence prediction, speech recognition, machine translation, spell/grammatical error correction

$P(w_{i+N} \mid w_i\, w_{i+1}\, w_{i+2} \ldots\, w_{i+N-1})$

Lec 3

Context-free grammar (CFG)

Noun phrase (NP) -> ProperNoun (专有名词)

NP -> Det Nominal

Nominal -> Noun

Det -> a

Det -> the

Noun -> flight

Can be used for: assigning a structure to a given sentence (parsing), and generating sentences

In Bi-Gram LM:

Maximum Likelihood Estimation (MLE):

$P(w_k \mid w_{k-1}) = c(w_{k-1}, w_k) / c(w_{k-1})$ (Bi-gram count/Uni-gram count)

Evaluation of LM: perplexity $PP(W) = P(w_1, w_2, \ldots, w_{N-1}, w_N)^{-1/N}$ lower PP->better

Training on 38m million words from WSJ and text on 1.5m shows tri-gram has lower perplexity than bi-gram

Problem 1: Data sparseness problem (some perfectly acceptable N-grams are mission due to finite training corpus) -> zero probability

Sol: assign non-zero to probability (N-Gram Model smoothing)

Discount: “steal” some probability from non-zero to zero ones

Un-smoothed Unigram: $P(W_k) = c(w_k) / N$

Add-One(Laplace) / k smoothed uni-gram: $P^*(w_k) = (c(w_k) +k) / (N+kV)$

Backoff: (lower-order N-gram has less severe issue compared to higher-order) if prob of N-gram is 0, then use (N-1)-gram instead.

Katz backoff: $P^*(w_k \mid w_{k-2}, w_{k-1}) = (c^*(w_{k-2}, w_{k-1}, w_k) / c(w_{k-2}, w_{k-1})$ if ~!=0 else $\alpha(w_{k-2}, w_{k-1})P^*(w_k \mid w_{k-1}))$

Stupid backoff web-scale N-grams: $P(w_k \mid w_{k-2}, w_{k-1}) = (c(w_{k-2}, w_{k-1}, w_k) / c(w_{k-2}, w_{k-1})$ if ~!=0 else $\lambda P(w_k \mid w_{k-1}))$

Linear Interpolation: mix the probability estimates. $P^ (w_k \mid w_{k-2,} w_{k-1}) = \lambda_1 P(\text{trigram}) + \lambda_2 P(\text{Bigram}) + \lambda_3 P(\text{unigram})$

Absolute discount: minus a fixed count 0.75 from the training set

Interpolated Kneser-Ney Smoothing: $P^(w_k \mid w_{k-1}) = \max(c(w_{k-1}, w_k)-d, 0)/c(w_{k-1}) + \lambda(w_{k-1})P_{CONTINUATION}(w_k)$, where $\lambda(w_{k-1}) = d/c(w_{k-1}) * \{w: c(w_{k-1}, w)>0\}$ (number of word types that can follow w_{k-1})

Katz backoff + smoothing: called a Good-Turing

Problem 2: Unknown words (out of vocabulary (OOV) problem

Closed (fixed) vocabulary: drop unknown words when encountered; Open vocabulary: special token <UNK> (need to create some <UNK> in the training data)

Two common ways: 1) choose a vocabulary and set any other words to <UNK>, 2) replace words with low frequency by <UNK>

Neural Language Model

N-Gram LM: Probability over word; Neural LM: Probability over word embedding

Better Generalization, Longer History, More Accurate, but, More complex, slower to train, less interpretable (AI interpretability)

FFN input: Hand-built features (wordcount, positive lexicon words, count of ‘no’), embedding of input words

Recurrent NN (sequential data) -> FFN, can be used for modelling sequence (e.g., part-of-speech tagging (sequence labelling), autoregressive generation (autoregressive: generate word one-by-one, language modelling (= Next word prediction)), sequence classification, encoder-decoder (two separate RNN models). variations: stacked RNN, bidirectional RNN

Encoder-Decoder Architecture (seq-to-seq model, RNN->context Vector->Decoder). Limitation: the hidden state of the last RNN in the encoder forms context vector so should contain all the information of the input, thus becomes bottleneck. Motivation: remember something in the input -> attention.

GPT is (transformer) decoder, only decoder can generate, encoder is for understanding

LLM

Prompt: Discrete (hard) (try different)/ Continuous (soft) Prompt (psudeo prompt -> prompt encoder -> input)

Prefix Tuning (specify task using specific prefix token), (param efficient) prompt tuning ()

Chain-of-thought (CoT) Prompting (few shot learning, In-context-learning, without fine-tuning): encourage the model to generate the thinking process, released the potential of reasoning capabilities to some extent.

Challenge: need to design both the original prompt and the decision process. -> Select some high quality CoT generated by the LLM.

Self-consistency: generate multiple times and choose the most frequent answer as the final answer.

Refinement / Correction with (self-) Feedback: LLM<--Refine&feedback→ model (evaluation)

CRITIC: self-correction with external tools (calculator, perspective text API, code interpreter, to verify and correct) -> tool learning (learn to cook / make PowerPoint)

Challenge: most times can only partially correct, sometimes fails to correct

RLHF (reinforcement learning with human feedback) -> RLAIF