WordNet

WordNet, a thesaurus containing lists of synonym sets and hypernyms ("is a" relationships). 【Great as a resource but missing nuance, e.g. "proficient" is listed as a synonym for "good". This is only correct in some contexts; Impossible to keep up-to-date;Subjective;Requires human labor; Can't compute accurate word similarity 】

Count based word representation

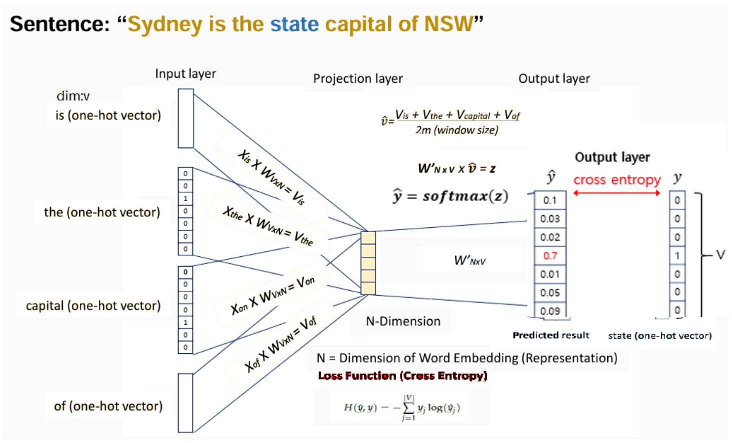
one-hot enc:: 【no word similarity representation, Inefficiency as most values are 0】 **bag of words**: The intuition is that documents are similar if they have similar content. table of words frequency, involves a vocabulary and A measure of the presence of known words. 【order or structure of words in the document is discarded, in turn meaning of words in the document (semantics) is discarded either】 **TF-IDF(inverse document frequency)**: how important a word is to a document. log is used to avoid weight explosion brought by big N to ease further calculation. df_i sometimes plus 1 to avoid zero division error and at least to make a very frequent word not having zero weight. $w_{i,j} = \mathrm{tf}_{i,j} * \log\left(\frac{N}{\mathrm{df}_i}\right)$ wij = weight of term i in document j; tfij = number of occurrences of term i in document j; N = total number of documents; dfi = number of documents containing term i.【slow for big vocab, need assumption that different counts of word brings different meaning, and ignoring semantic similarities between words】

Prediction based word representation【cannot cover morphological similarity; with rare words, cannot be trained well(underfitting); cannot handle OOV, no way at all; Training dataset reflect the word representation result】

word vector similarities brought by cosine.

1. word2vec, 2 training way:
   - Continuous Bag of Words(CBOW)

   V is the dim of one-hot representation of word N is the target embedding dim


Sentence: "Sydney is the state capital of NSW"
Input layer
Projection layer
Output layer
dim:v
is (one-hot vector)
the (one-hot vector)
capital (one-hot vector)
of (one-hot vector)
$X_{is} \times W_{V\times N} = V_{is}$
$X_{the} \times W_{V\times N} = V_{the}$
$X_{on} \times W_{V\times N} = V_{on}$
$X_{of} \times W_{V\times N} = V_{of}$
$\hat{v} = \frac{V_{is} + V_{the} + V_{capital} + V_{of}}{2m \text{ (window size)}}$
$W'_{N\times V} \times \hat{v} = z$
$\hat{y} = softmax(z)$
$W'_{N\times V}$
N-Dimension
Output layer
$\hat{y}$
cross entropy
y
0.1
0.03
0.02
0.7
0.01
0.05
0.09
V
Predicted result
state (one-hot vector)
N = Dimension of Word Embedding (Representation)
Loss Function (Cross Entropy)
$H(\hat{y}, y) = -\sum_{j=1}^{|V|} y_j \log(\hat{y}_j)$

   - Skip-Gram
   - windows size matters. Smaller window sizes (2-15) lead to embeddings where high similarity scores between two embeddings indicates that the words are interchangeable. Larger window sizes (15-50, or even more) lead to embeddings where similarity is more indicative of relatedness of the word
   - Negative Samples matters Negative samples to our dataset. samples of words that are not neighbors. The "negative samples" are selected using a "unigram distribution", where more frequent words are more likely to be selected as negative samples. The probability for picking the word (*wi*) would be equal to the number of times (*wi*) appears in the corpus, divided the total number of word occurs in the corpus.

]

1. FastText(still use CBOW or Skip-gram to train) use n-gram to seperate words, therefore rare words and OOV words and morph words may have existing parts in training set. word embedding is the average or its character parted embeddings.
2. Glove Use co-occurance to obtain global stat features instead of just local context.

Evaluation on word vectors

try different training methods(CBOW, skip-gram...), dimensions, window size, try to get highest acc on semantic tests(Beijing Tokyo), syntactic tests(great greater), on different datasets.

1. Intrinsic, analogy task, find the most similar vector in vocab.(using cosine) 【fast, may not be really helpful to real task unless strong connection is proved, as tests are human designed】
2. Extrinsic, apply word vec to real tasks. kinda like end2end. 【taking time, real systems may involve other factors.】

Deep Netual Network

Activation Function aims to add a non-linear property to the function. The softmax will enforce that the sum of the probabilities of your output classes are equal to one. It transforms a vector into a vector whose i-th component is:

$$\frac{e^{\widehat{y}_i}}{\sum_{j=1}^{|V|} e^{\widehat{y}_j}}$$

【1.Exponentiate to make positive; 2.Dividing by $\sum_{j=1}^{|V|} e^{\widehat{y}_j}$ normalizes the vector $\sum_{j=1}^{n} \widehat{y}_j = 1$ to give probability】

Recurrent Netural Network

- vanilla RNN

$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$ The tanh activation is used to help regulate the values flowing through the network. The tanh function squishes values to always be between −1 and 1. And, for large values will be extremely large, small values will be extremely small. parameter are shared in same layer, in order to handle variant length of input. (very long sequence)Use Truncate Backpropagation through time, to update weight by gradient from a certain range of timestamps. so loss_5 is to update all first five, then loss_10 is to update 6 to 10. only works n-to-n. 【Vanishing Gradient Issue, neurons in early layers learn very slowly】 or【Exploding Gradient Issue, large updates, not stable, may NaN in the end】 when multiply W_{hh} too many times

- Long Short Term Memory(LSTM), mitigate gradient vanishing problem through gating. by using sigmoid as activation, small values wouldn't be Negative as using tanh, instead will have a really close to zero value. so small values will be forget.

$\text{forget}_t = \sigma\left(W_{f[h_{t-1},x_t]} + b_f\right)$

$\text{input}_t = \sigma\left(W_{i[h_{t-1},x_t]} + b_i\right)$

candidate : $\widetilde{C}_t = \tanh\left(W_{c[h_{t-1},x_t]} + b_c\right)$

below is to apply different weights to the last time stamp and the current one. $\text{Cell}_t = f_t * C_{t-1} + i_t * \widetilde{C}_t$; and go through forget and tanh again, boring. $O_t = \sigma\left(W_{o[h_{t-1},x_t]} + b_o\right)$

$h_t = O_t * \tanh(C_t)$

- Gated Recurrent Unit (GRU), good for small short dataset, but failed to LSTM when facing long sequence.

data transformation

concatenation, most used. summation(element-wise summation), efficent as lower dimension

Application: Sequential Text Classification

bi-rnn n to 1, loop from both directions, then final state is the concatenation(or summation) of each directions' last state.【summation may information leakage(you cannot distinguish which part of the vector is what as you are facing a merged vector) 】

- sentiment classification

chanllenge: Sarcasm(反讽),Word Ambiguity(word "unpredictable" may express totally different emotion in different context),Multipolarity how to solve: find the target object, its attributes(components, properties), attitude holder, type of attitude(positive, negative), time

Sequential models

is the task of predicting what word comes next based on the given word is a probabilistic model which predicts the probability that a sequence of tokens belongs to a language.

1. Traditional Neural Language model【Pros: No Trade-off issue;】【Cons: Window size selection issue (increasing window size enlarges W); Input vectors are multiplied by completely different weights in W (No symmetry in how the inputs are processed)】
2. RNN-based Language Model【Pro: Can process any length input; Can use information from many step back; Model size does not increase; Same weights applied on every time step (Symmetry)】【Cons: Slow computation; Difficult to access information from many step back】

Teacher Forcing: During training, we feed the gold (aka reference) target, regardless of what each cell predicts. This training method is called Teacher Forcing.

language fundamental

phonology(voice) => morphology(by lexical, the structure of words) => syntax(grammer parsing, the way words are used to form phrases) => semantics(contextual reasoning, Compositional and lexical semantics) => reasoning(application level with domain knowledge) Morphemes are the pieces of words: bases, roots and affixes (pre-fix, suffix). Words in a language consist of one element or elements of meaning which are morphemes

Text Preprocessing

Type: an element of the vocabulary. Token: an instance of that type in running text Tokenization is facing language issues, some lang has words with too much meaning like germany, some lang contain multiple alphabets. some lang like arabic contains words written in different orders.

words normalization is needed(U.S.A and USA); case folding(careful when US and us)

Lemmatisation, Reduce inflections or variant forms to base form • am, are, is à be • car, cars, car's, cars' à car

Stemming Reduce terms to their stems in information retrieval e.g., automate(s), automatic, automation all reduced to automat

	Stemming	Lemmatization
Efficiency	FasterIt chops words without knowing the context of the word in given sentences.	Slower, compared to stemming.It knows the context of the word before proceeding.
Approach	Rule-based Approach	Dictionary-based Approach
ExpectedOutcome	When we convert any word into root form then stemming may create a word with no meaning.	Lemmatization always gives the dictionary meaning while converting words into root form.
Usage	It is used when the meaning of the word is not important.e.g. Spam Detection	When it is important to retain the meaning of the word (context)e.g. Question Answering

Sentence Segmentation, use regex.

Part-of-Speech Tagging(POS)

A class of words based on the word's function, the way it works in a sentence. Essential ingredient in natural language applications larger and more fine-grained tag sets are preferred.Trade-off between complexity and precision and whatever tag-set we use, there will be some words that are hard to classify. Criteria:

- Distributional, Where can the words occur? like Adverb may appear at the end of a sentence.
- Morphological, What form does the word have? What affixes can it take?
- Notional(semantic) What sort of concept does the word refer to? (nouns often refer for people,places)

Issue: Tag sets are quite counterintuitive

Useful:【in many tasks like Text-to-speech, Lemmatization, Linguistically motivated word clustering; as a pre-processing step for parsing; as features to downstream systems.]

Baseline Approaches:

- Rule-based Tagging, Old way: first identifies a set of possible POS for each word in the sentence (based on a lexicon), and the second uses a set of hand-crafted rules in order to select a POS from each of the lists for each word. Assign each token all its possible tags. Apply rules that eliminate all tags for a token that are inconsistent with its context. Assign any unknown word tokens a tag that is consistent with its context(e.g. the most frequent tag)
- Lookup Table, based on history stat data of how a word be assigned. We will use the most frequent POS of this word.
- N-gram. Still use lookup table, but considered as a combo of neighbored words, more accurate. OOV? no fancy solution, fallback to above methods

– Seq2Seq Approaches: bi-lstm, can also be with word embedding concat character embedding.

Name-Entity locate and classify named entity mentions in unstructured text into pre-defined categories NER performance? Precision(Detected as 'PERSON' correctl / Total number of detected as 'PERSON') and recall(Detected as 'PERSON' correctly / Total number of actual 'PERSON' entitie) $F_1 = 2 * \frac{P*R}{P+R}$ The IOB (short for inside, outside, beginning) is a common tagging format. Compare with IO tag, more time may be need as more tags exist in IOB. and IO cannot know if a muplti-word entity exists. dependency between entities may be learned by adding an extra layer of sequence model

Slot Tagging(used in Spoken Dialog System)
1. find all slots(like NER tagging), 2. intent classification

Question Answering

Questions can have different types: general questions, wh-questions, choice questions, and factoid questions(answer is inside a piece of text) Answer format? where can I find answer? trianing data format? Approches:

- Knowledge-based QA (Semantic Parsing), parse question to a structed query, so we obtain answer from a predefined database. 【require experts to annotate and maintain, and hard to find a well-structured training dataset. Constrained to queriable questions in Database Schema】【system is robust and answer is from trustable sources, Answer independent of question and parsing mechanism】. seq2seq model could help to do semantic parsing(trianing labels would be structure form of a query)
- Information Retrieval-based QA (Reading Comprehension), answer is by finding sentence or paragraph or documents, answer by quote. document and question need to be sent to two different network, then use attention between them to get the related part of the document.
- Visual QA, given an image as context(instead of document), ask something, need to answer in text. Idea is similar to reading Comprehension, except CNN is used to extract image features and attention is being calculated between CNN extracted features and embedded/LSTMed question vector.
- Open-domain QA, back to textual. Instead of given a document as context in runtime, Open-domain QA is trained with a large collection of documents and remembered it. you just need to ask. LLM could do this task too, but the reference may be a fake one, see LLM hallucination.

Attention

in seq2seq, the last hidden state of encoder is everything that a decoder need, but suffering from vanishing gradient when text is too long. Attention could help that. Direct connection is established between decoder's hidden to every timestamp's hidden of encoder, using dot product, then softmax is applid to obtain a probability distribution over all encoder inputs, giving the highest probability to the timestamp that is most important. Then a weighed sum of all encoder hiddens will be done, this also contains every encoder hidden's information but the most important one will be more significant. concat with the last hidden step, you got a vector that twice the hidden size, doing classification task or other final task output on that please.

【Benefit? Improve performance by Allow decoder to focus on certain parts of the source; Solving the bottleneck problem by Allow decoder to directly look at the source (input); Reducing vanishing gradient problem by Provide shortcut to faraway states; Providing some interpretability by Inspect attention distribution, and show what the decoder was focusing on】

Global: calculate attention over all sequence Local: calculate attention over part of sequence

self-attention: target sequence is itself

Machine Translation

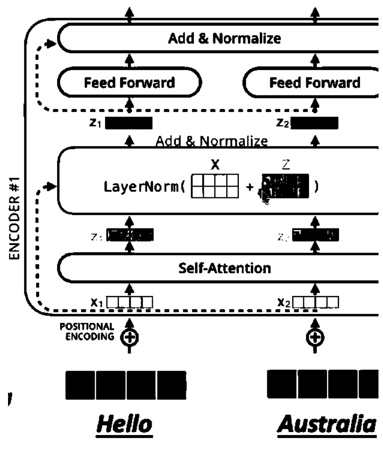
1. Statistical Machine Translation

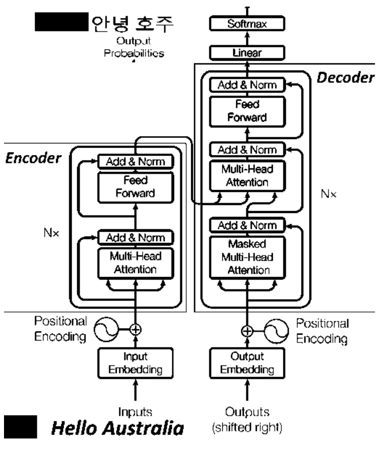
Learning a probabilistic model from data $\text{argmax}_y P(x|y) * P(y)$ former part models how words are translated to target language, and latter models how fluent the target sentence is.
First use parallel corpus to train a model that output borken target language, $P(X|y) = P(x, a|y)$, a is Alignment, which is the correspondence between particular words in the translated sentence pair. (i.e. word-level correspondence between source sentence x and target sentence y)

Then use target Corpus and Language Model to regular。(with all possible words we have from last step, give a sequence that is having highest probability, based on last timestamp prediction, RNN is good for this)

1. Netrual Machine Translation

Transformer





Use self-attention in the encoder, instead of RNN or CNNs Predict each translated word Final cost/error function à standard cross-entropy error on top of a softmax classifier

6 layers encoder, 6 layers decoder; encoder goes layer by layer, last layer's encoder output will go every decoder's input. decoder also layer by layer, but each layer is having same encoder output. embedding dimension 512, number of heads 8.
1. positional embedding

need to have same identifier no matter how length of input changes. and cannot use self increment integer as it will have very large values dominate the word vector. $\text{PE}_{\text{pos},2i} = \sin\left(\frac{\text{pos}}{10000^{2\frac{i}{d_\text{model}}}}\right)$ $\text{PE}_{\text{pos},2i+1} = \cos\left(\frac{\text{pos}}{10000^{2\frac{i}{d_\text{model}}}}\right)$ summation is used to integrate with word embeddings

1. multihead-attention,try to get a different view of input, then concat, and goes linear to reduce dimension.
2. Add & Norm, Residual connection is used. we add input embedding and self-attention output, this sum will be then normalized. this is to prevent lose information from self-attention.

Natural Language Generation

Conditional Language Modeling: the task of predicting the next word, given the words so far, and also some other input x: "P(An, adorable, little, boy, is, spreading, smiles)= P(An) × P(adorable|An) × P(little|An adorable) × P(boy|An adorable little) × P(is|An adorable little boy) × P(spreading|An adorable little boy is) × P(smiles|An adorable little boy is spreading)" P(is|An adorable little boy) = Count(An adorable little boy is)/Count(An adorable little boy)

to avoid no 'An adorable little boy is' phrase in corpus, we can use n-gram, predict next word only based on n words before. Zero Count Issue? P(w|is spreading) = Count(is spreading w) / Count(is spreading) What if the 'is spreading w' phrase never occurred in the corpus Alternative solution: Smoothing (Add small � to the count for every w in the corpus)

What if the 'is spreading' phrase never occurred in the corpus? It is impossible to calculate the probability for any w. • Alternative solution: Backoff (Just condition on "spreading" instead)

Traditional way is relying on the choice of window size, and suffer from different weights for different word.

RNN model using seq2seq is facing two ways of generating words:
1. greedy decoding, just decode the word until "" appears. 【No way to back tracking, easily breakout when only one prediction is wrong】
2. Beam Seach, find sequence with highest probability, pick top-k(k is beam size) output(among ALL CHOICEs) each time, you should always track K sequence. sequence score is given $\sum_{i=1}^{t} \log P_{\text{LM}}(y_i \mid y_1, \cdots, y_{i-1}, x)$【taking computational resource; In open-ended tasks like chit-chat dialogue, large k can make output more generic】

pretrained models

use pretrained weights instead of randomly initiatlize.

1. byte-pair-encoding (BPE) all words are split into characters, find the character pair that has the most frequency, merge, then repeat this process. until designed vocab size or 10th round. e.g: l o w e s t => l o w es t => l o w est , this way "est" will be found.

2. contextual representation; In different context, word could have different meaning. different embedding should be used. Need train embedding together.

Pretraining methods

first motivation: get contextual representation
1. ELMo, bidirectional-lstm

GPT, unidirectional-transformer, Remove ENcoder and related multihead attention in decoder part, only masked-multihead-attention, only left -to right due to masked. **BERT**, bi-transformer, using encoder; token embedding, segment embedding and position embedding make it able to handle different types of tasks. two pre-training task:1.using mask, to enable pretrain; predict masked word; 2.next sentence prediction, learning relationships between sentences **RoBERTa**: fine-tuned and more data version of BERT **XLNet**, relative position encoding, how much a word should attend to the previous word; permute word order during self attention **ALBERT**, factorlize embedding parameter, break a big matrix into two smaller one.(light, but slow), share all parameters between transformer layers **ELECTERA**, regard mask word prediction as discrimanaion, label the word which is replaced.

GPT!GPT!GPT!

1. GPT, GPT-2 add Layer Norm before attention, feed-forward and linear output.

training data, parameters, layers and dimensions are all increased when GPT-2 and GPT-3 came out.GPT-3 also uses Sparse Attention(will skip some tokens). ChatGPT is different, it adds Reinforcement Learning with Human feedback and safety control.

1. LLaMa, larger dataset, smaller model, training longer time, can be train on a small GPU, opened. using RMSNorm (no weight matrix re-centering), using Swish+GLU instead of ReLu, using Rotary Embedding(absolute and relative) instead of absolute positional embedding

2. Alpaca, similar performance as GPT-3.5, but smaller model. first use human created 175 instruction-output pairs as seed sets. use gpt-3.5 to generate more instructions using seed sets. (cost less than 500$); generate 52K unique pairs, use them to fine-tune LLaMa, then got Alpaca. As 175 instructions are not too many, and most work could be done by using ChatGPT(in a not expensive way), they call Alpaca a Replicable model

3. Vucuna, use GPT-4 as assessment(score), cheaper, can run on CPU

Problems（observe large corpus of text, find pattern, and just mimic.

1. too general, lack domain knowledge
2. cannot modify
3. hallucination, inherent, the output contradicts the input logically or external, given input is not suffice to veryify the output's accuracy. possible solution may be using a chain-of-thought output(taught model during training)
4. safety concerns like: indistinguishability between generated and human-written text may cause malicious purpose events. using watermark(a special hidden pattern of text)
5. limited Context length
6. planning and reasoning