Language Models – N-gram, Neural Network for Language Models, Transformer based models

Introduction

Language Models are pivotal to Text Mining and Analysis and are the backbone of some of the modern use cases for Natural Language Processing like, Speech Recognition, Language Translation, Question Answers, Sentiment Analysis etc. It is a model that helps to predict next word based on preceding words, text summarization, generating completely new pieces of text. It takes a lot of text data which is relevant to train such a model and effectively use in practical use cases. The evolution of LMs have taken place from most basic, Statistical n-gram Language Models to Neural language modes (the window based and RNNs) relying on sequence of words, further to more modern Transformer based models like BERT combined with multi-task learning. In this review we'll talk about them and how they mitigate shortcomings of the previous ones.

N-gram Language Models

The most fundamental is the n-gram language model, with the basic idea to collect statistics about how frequent different n-grams are and use these to predict next word. That is to identify the word probability distribution of the previous words (in history) and predict the next word. They are based on the Markov's assumption that the probability of a word depends only on the previous word.
Markov models are the class of probabilistic models that assume that we can predict the probability of some future unit without looking too far in the past.
Statistical Language models which assign probabilities to a sequence s of N words, i.e.,

$$P(s) = P(w_1 w_2 \cdots w_N)$$
$$= P(w_1)P(w_2|w_1)\cdots P(w_N|w_1 w_2 \cdots w_{N-1})$$

N-grams are considered as sequence of N-words, and based on that

a) 0-gram is a zero-gram model where all words from the vocabulary (V) have equal probability

$$p(w_i) = \frac{1}{|V|}$$

b) 1-gram is unigram-model which using maximum likelihood estimation and probabilities are based on word counts

$$p(w_i) = \frac{count(w_i)}{count(w)}$$

c) 2-gram (or bigram) is sequence of two words using maximum likelihood estimation, probabilities are based on bigram and word counts

$$p(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

d) 3-gram (or trigram), a sequence of 3 words using longer continuous history

$$p(w_i|w_{i-2}, w_{i-1}) = \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})}$$

e) For N-gram, current state depends upon previous N-1 states

$$p(w_i|w_1, w_2...., w_{i-1}) \approx p(w_i|w_{i-(N-1)} \cdots w_{i-1}) = \frac{count(w_{i-(N-1)}, \ldots w_{i-1}, w_i)}{count(w_{i-(N-1)}, \ldots w_{i-1})}$$

Advantages:
It is a very efficient technique for inferring, since, as soon as the counts have been computed, probabilities are output in O(1) time using lookup tables.

Challenges:
It suffers problem of sparsity and assumes a closed vocabulary size known advance and there are no new words – which is unlikely in practical scenarios. If the numerator has unknown words then probability will be '0', whereas if the denominator has unknown words, the probability will be undefined.

It could be mitigated to some extent by adding La Place smoothening, so that there is no zero probability of unknown words, as follows

$$p(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i) + 1}{count(w_{i-1}) + |V|}$$

Even with this there is still a curse of dimensionality. N-gram models is very sensitive to the Training Corpus size, the number of unique N-gram grows exponentially with N. For example, if one wants to model an n-gram LM with a vocabulary of size 10, 000, there are potentially $10000^n - 1$ free parameters.

Neural networks for Language Modeling

Neural Networks for Language modeling includes Feed Forward Neural Network (FFNN) and Recurrent Neural Network (RNN), that automatically learn features and continuous representation. Smoothing is solved implicitly. It uses back propagation for training.

Feed Forward Neural Network (FFNN)

The low dimensional **window-based neural language model**, solve the problem of sparsity and scale in the continuous space, where similar words are automatically clustered together.
It represents words as vectors (word embeddings), and inputs them to the neural language model. These word embeddings are concatenated in the projection layer and fed into a hidden

layer, whose output is then provided to a softmax layer, that produces a probability distribution over words in vocabulary, V.

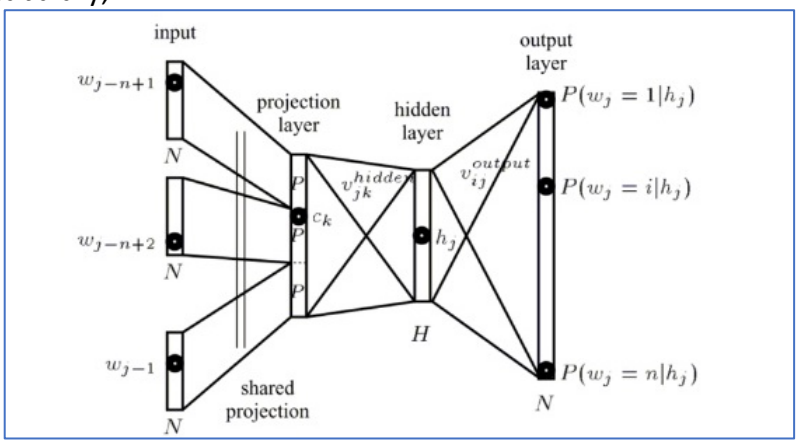

Figure: A neural language model (Bengio et al., 2006)

The model tries to maximize the average of the log probabilities of all words in the corpus given their previous n words:

$$J_\theta = \frac{1}{T} \sum_{t=1}^{T} \log p(w_t | w_{t-1}, \cdots, w_{t-n+1})$$

$$= \frac{1}{T} \sum_{t=1}^{T} \log f(w_t, w_{t-1}, \cdots, w_{t-n+1})$$

f(wt,wt−1,⋯,wt−n+1) is the output of the model, i.e. the probability p(wt|wt−1,⋯,wt−n+1)p(wt|wt−1,⋯,wt−n+1) as computed by the softmax, where *n* is the number of previous words fed into the model.

Word2Vec

Many word embedding models were created but the challenge was high computational cost for softmax layer over a large vocabulary V.
The most promising one was Word2Vec, that eliminated the expensive Hidden Layer and enable the LM to take additional context into account.

Enhancing the training strategies like

a) Continuous bag-of-words (**CBOW**), which instead of feeding n previous words into the model, it feeds a window of n words around the target word wt at each time step t

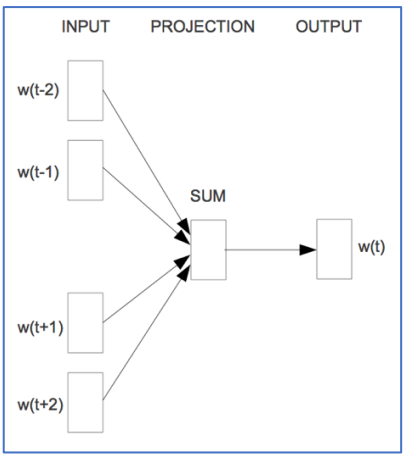

Figure : Continuous bag-of-words (Mikolov et al., 2013)

$$J_\theta = \frac{1}{T} \sum_{t=1}^{T} \log p(w_t | w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n})$$

b) **Skip-gram**, uses the center word to predict the surrounding words. It sums the log probabilities of the surrounding n words to the left and to the right of the target word w_t

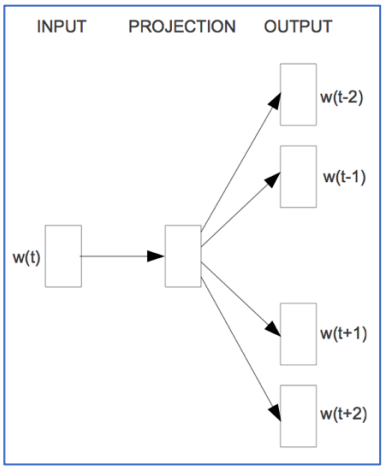

Figure: Skip-gram (Mikolov et al., 2013)

$$J\theta = \frac{1}{T} \sum_{t=1}^{T} \sum_{-n \le j \le n, \neq 0.} \log p(w_{t+j} \mid \mathbf{w_t})$$

Recurrent Neural Network (RNN)

The RNN perform the same task for every element of a sequence, with the output depended on previous computations. It provides more structure and parameter sharing. In theory they use info from long sequences with the output dependent on previous computations. This enabled

RNN based language models to process sequence data as variable length and share parameters across internal states.

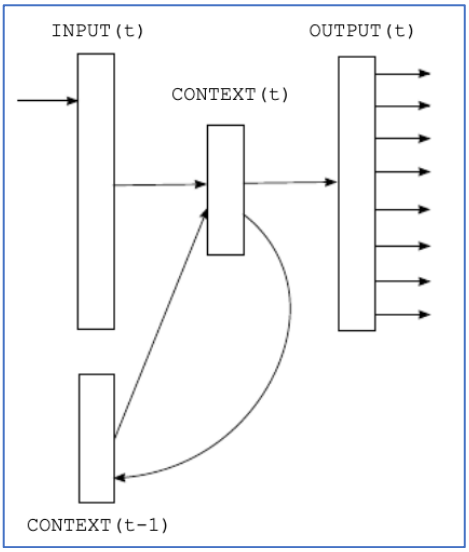

Figure: Recurrent neural network-based LM

The recurrent network has an input layer x, hidden layer s(also called context layer or state) and output layer y.
Input vector x(t)is formed by concatenating vector w, representing current word, and output from neurons in context layer *s* at any given time *t−1*.
To improve performance, infrequent words are usually merged into one token.

RNN Model

$x(t) = w(t) + s(t-1)$
$s_j(t) = f\left(\sum^{X}_{i} x_i(t)\, u_{ji}\right)$
$y_k(t) = g\left(\sum^{X}_{j} s_j(t)\, v_{kj}\right)$

where f(z) is sigmoid activation function:

$$f(z) = \frac{1}{1 + e^{-z}} .$$

and g(z) is softmax function:

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}$$

Challenges:
A drawback of RNN based LMs is that they rely on sequential modelling that are hard to parallelize. They also do not learn long term dependencies even with LTSM/GRU improvements.

Transformer based Language Models

Transformers are multi-headed attention based self-supervised architecture and provides context to the word vectors.
Transformer based LMs embed context to the words. This is an improvement over Word2Vec. In addition to applying the context, it also provides the positional encoding vector, so that the network knows the word order. The residual connection between the sublayers, ensure better flow of information through the network.
Example:
What is the fair prize of the share?
I enjoyed at the fair.
BERT will generate different word vectors for both instances of the word “fair”
Word2Vec will compute equal value word vectors.

Transformers take a step beyond just text-based computations, it can process multi modal inputs (text and images) simultaneously.

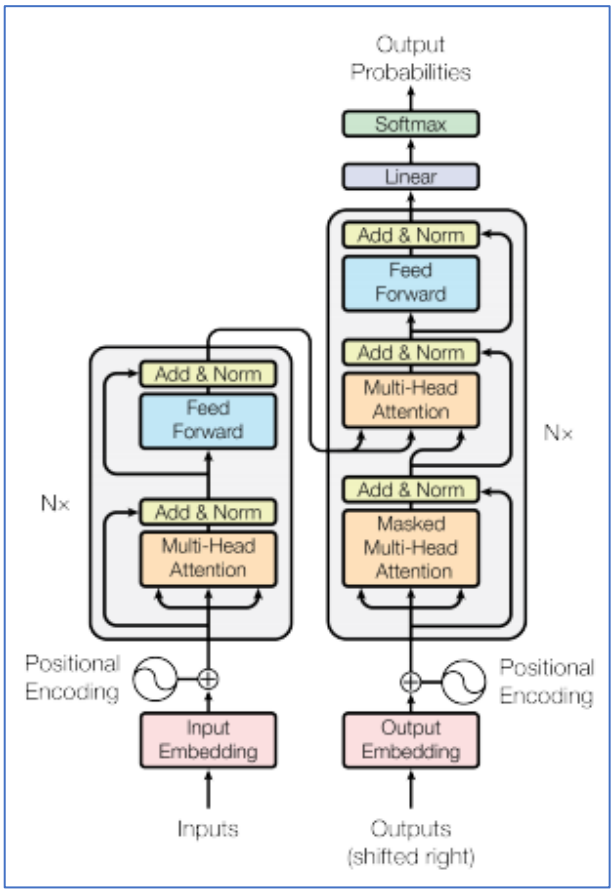

Figure: The Transformer - model architecture from (Attention is all you need)

Main components of Transformer based architecture are:

a) Encoder decoder stack
b) The attention functions which can be Scaled dot product attention or Multi head attention
c) Position wise feed forward networks
d) Embeddings and softmax for final output
e) Positional encoding

There are many Transformers based modern language models - BERT & GPT being the most known recent ones.
BERT, by Google API, provides bidirectional transformer with training objectives of "Masked language modelling" and next sentence prediction.

Conclusion

The evolution of Language Models has enabled us to approach the use cases like Automatic Speech Recognition, Conversations AI, Sentiment analysis, Text generation, Digital Assistants and many more, in an optimal way. A lot of intelligence and common sense is being developed with the help of these models, so much so that the machines are treading the path of unsupervised learning. With combination of multitask learning and transferring knowledge across different domain the language models will be worked upon performance to make the systems experts in their functioning.

References:

1) https://jon.dehdari.org/tutorials/lm_overview.pdf
2) Language Models: N-Gram - A step into statistical language modeling - Shashank Kapadia [https://towardsdatascience.com/introduction-to-language-models-n-gram-e323081503d9
3) A Survey on Neural Network Language Models Kun Jing∗ and Jungang Xu∗ [https://arxiv.org/pdf/1906.03591.pdf]
4) Mikolov, Tomas & Karafiát, Martin & Burget, Lukas & Cernocký, Jan & Khudanpur, Sanjeev. (2010). Recurrent neural network based language model. Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010. 2. 1045-1048.
5) Sebastian Ruder, "On word embeddings - Part 1". http://ruder.io/word-embeddings-1/, 2016.
6) Recurrent Neural Networks: The Powerhouse of Language Modeling - James Le [https://builtin.com/data-science/recurrent-neural-networks-powerhouse-language-modeling]
7) RNN – Language Modelling and Sampling Novel Sequences Strahinja Zivkovic
8) Attention Is All You Need [https://arxiv.org/pdf/1706.03762.pdf]
9) How do Transformers Work in NLP? A Guide to the Latest State-of-the-Art Models https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/
10) https://blog.feedly.com/nlp-breakfast-2-the-rise-of-language-models/