# 1，Distributed Representations of Words and Phrases and their Compositionality

## What problems have been solved in this paper ?

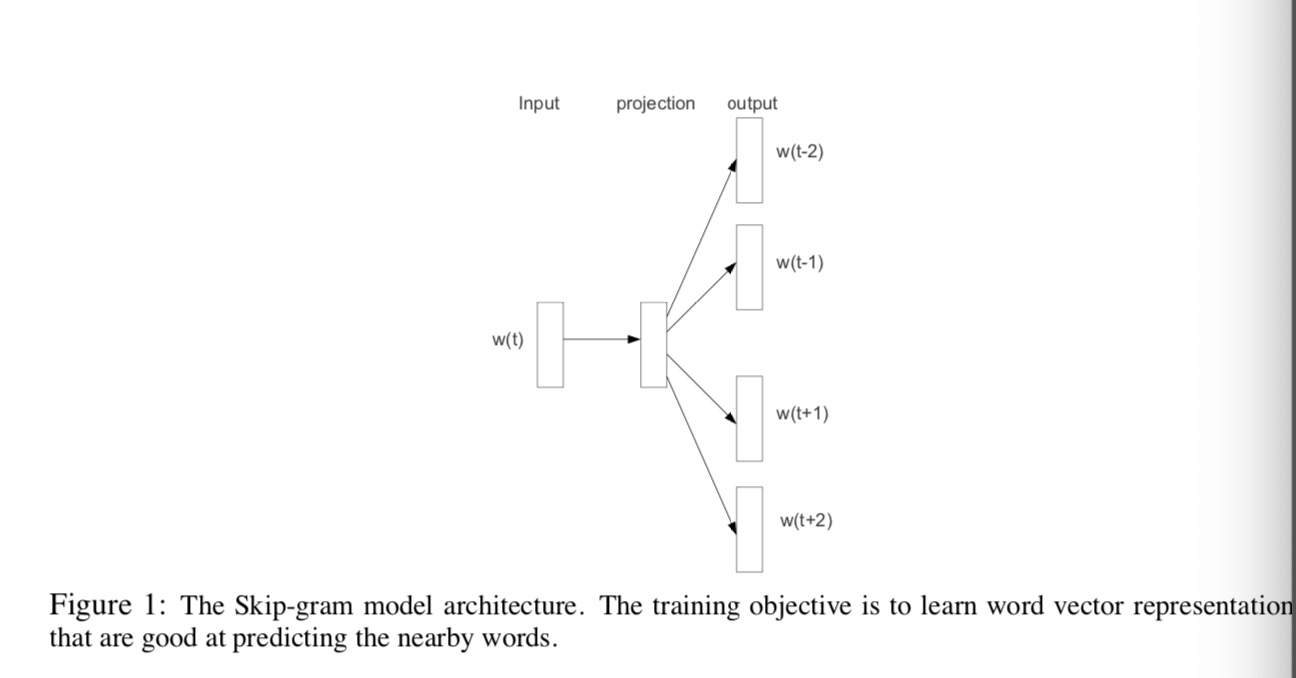
# Present several extensions of original Skip-gram model.

# 1, Use subsampling of frequent words during training to speedup the training and improve accuracy of the representations of less frequent words.

# 2, Present a simplified variant of Noise Contrastive Estimation(NCE) for training the Skip-gram model that results in faster training and better vector representations for frequent words, compared to complex hierarchical softmax that was used in the prior work.

# 3, Extend from word based to phrase base models. As word representations are limited by their inability to represent idiomatic phrases that are not compositions of the individual words.

## Model architecture



# Given a sequence of training words , the objective of skip-gram model is to maximize the average log probability

# *c is size of training context; wt is the center word.*

# The basic Skip-gram formulation defines using softmax function:

# are the ‘input’ and ‘output’ vector of w, and W is the number of words in the vocabulary.

## Breakthrough components

# 1, Extension from word based to phased based models

# Identify a large number of phrases using a data-driven approach, and treat the phrases as individual tokens during the training.

# To evaluate the quality of the phrase vectors, a test set of analogical reasoning tasks that contains both words and phrases was developed. For instance, *test sets “Montreal”:“MontrealCanadiens”::“Toronto”:“Toronto Maple Leafs”, It is considered to have been answered correctly if the nearest representation to vec(“MontrealCanadiens”) – vec(“Montreal”)+vec(“Toronto”) is vec(“Toronto Maple Leafs”)*

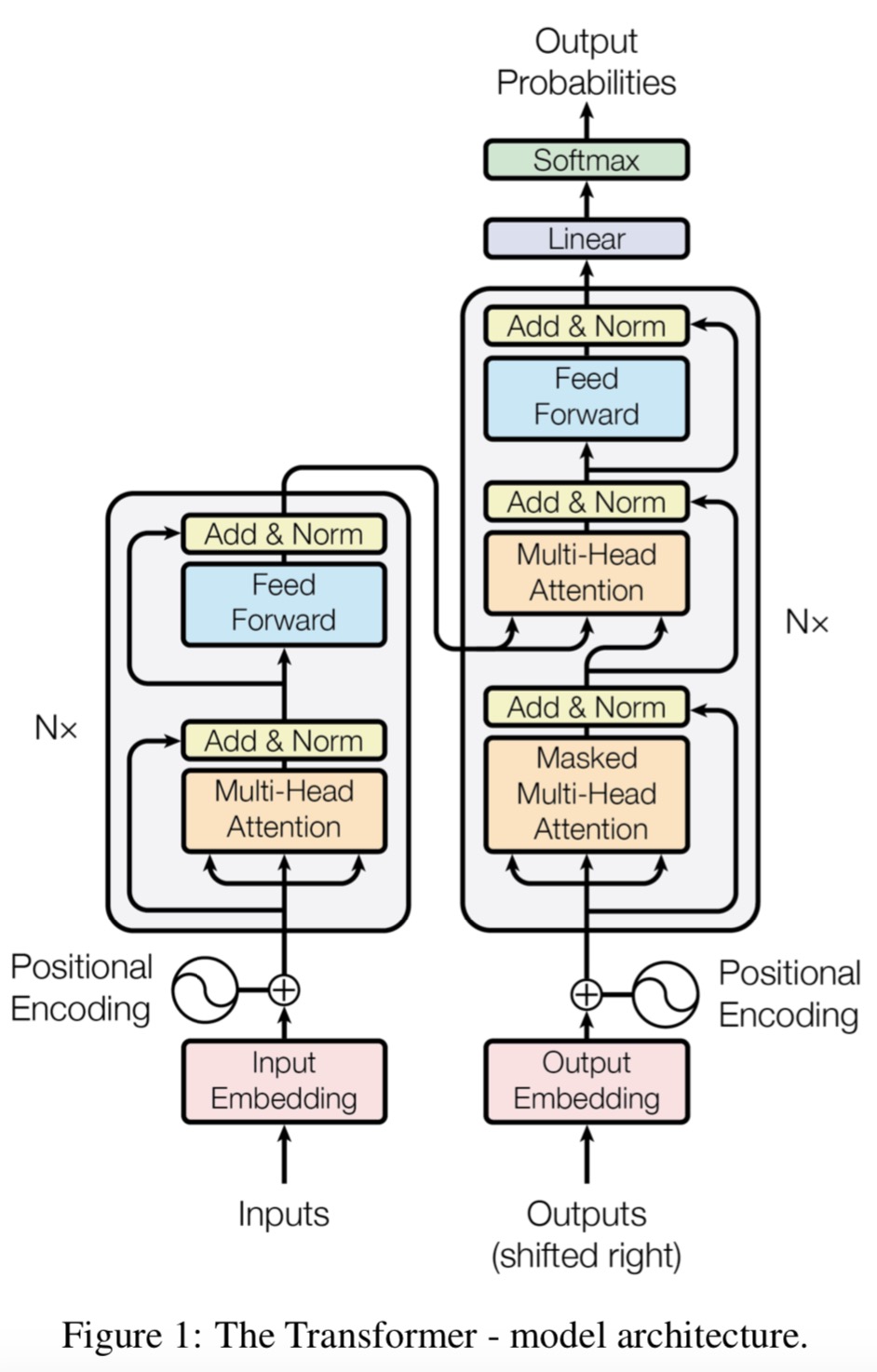
# 2，Attention Is All You Need

## What problems have been solved in this paper ?

# Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states ht, as a function of the previous hidden state h(t-1) and the input for position t. This inherently sequential nature precludes parallelization within training examples

## Model architecture

* **Overall architecture**: encoder- decoder structure

****

* **Encoder architecture**

Encoder is composed of a stack of N=6 identical layers.

* **Each identical layer for two layers**

**1st layer :**

1. Multi-head self-attention layer + residual connection + layer normalization;
2. ***A self-attention layer***: all of the keys, values and queries come from the same place, in this case, the output of previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
3. Output of 1st layer = LayerNorm( x+ multi-head\_layer(x))

**2nd layer:**

1. Position-wise fully connected feed-forward layer + residual connection + layer normalization;
2. Output of 2nd layer = LayerNorm( x+ feed-forward\_layer(x))

* **Decoder architecture**

Decoder is composed of a stack of N=6 identical layers.

* **Each identical layer for three layers**

**1st layer:**

1. Masked Multi-head self-attention layer + residual connection + layer normalization. ***For the masked multi-head self-attention, aim to prevent positions from attending to subsequent positions, ensure the predictions for position i can only depend i can depend only to the known outputs at positions less than i.***
2. ***A self-attention layer:*** allow each position in the decoder to attend to all positions in the decoder up to and including that position. In order to prevent leftward information flow in the decoder to preserve the auto-regressive property, implement this inside of scaled dot-product attention by masking out(setting to ) all values in the input of the softmax which correspond to illegal connection.
3. Output of 1st layer = LayerNorm( x+ masked\_multi-head\_layer(x))

**2nd layer – encoder-decoder layer:**

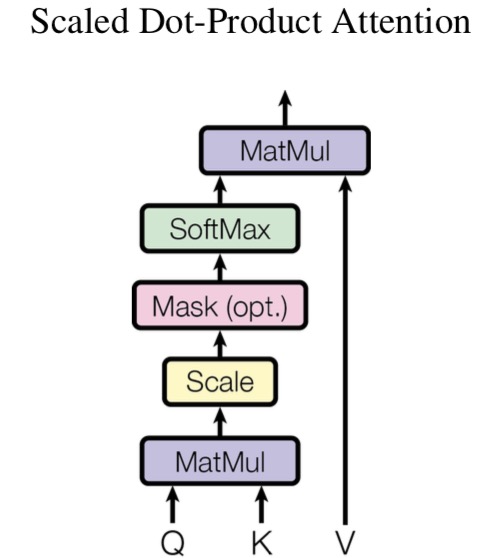
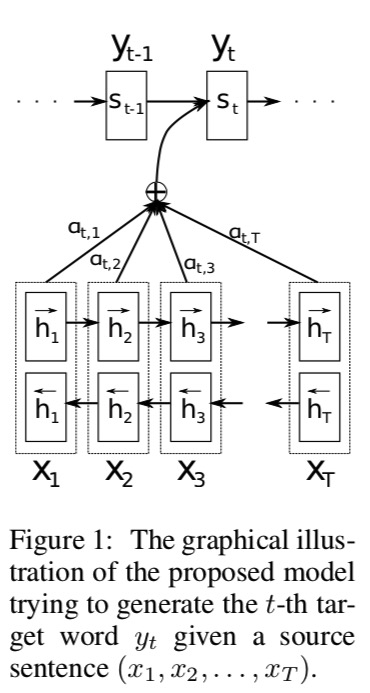
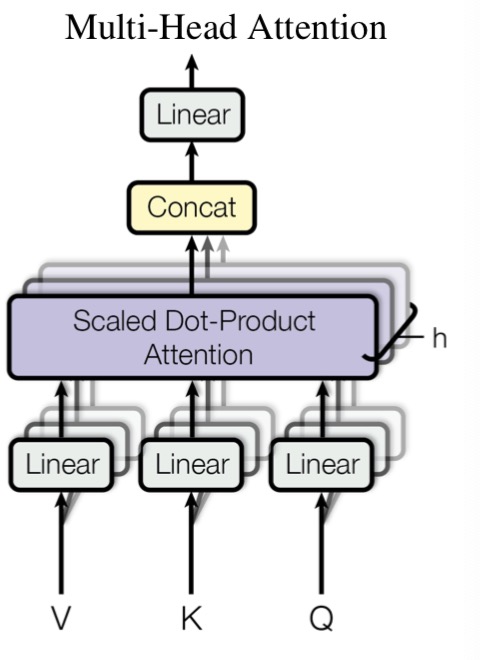
1. Multi-head self-attention layer + residual connection + layer normalization
2. The queries come from previous decoder layer; The memory keys and values come from the output of the encoder. This enable every position in the decoder to attend over all positions in the input sequence. This part mimics the typical encoder-decoder attention mechanisms in seq2seq models.
3. Output of 2nd layer = LayerNorm( x+ multi-head\_layer(x))

**3rd layer:**

1. Position-wise fully connected feed-forward layer + residual connection + layer normalization;
2. Output of 2nd layer = LayerNorm( x+ feed-forward\_layer(x))

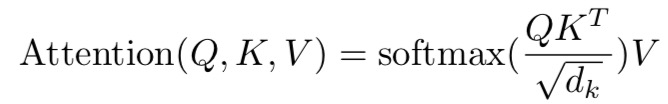
* **Attention**

Attention function: mapping a query and a set of key-value pairs to an output, where the query, keys, values and output are all vectors.

**** ****  

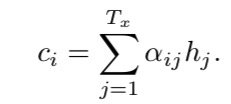
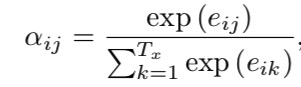
* **Scaled Dot-Product Attention**

Input:queries with dimensions dk , values with dimensions dv

Function: ****

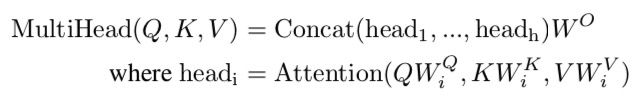
* **Additive Attention *–*** *not used in Transformers*

Computes the computes the compatibility function using a feed-forward network and a single hidden layer. Additive attention scores how well the inputs around position j and output at position i match. The score is based on the RNN hidden state and the j-th hj of the input sentence.

1. si is an hidden state for time i,  computed by ci is the context vector for target word yi
2. ci is computed as weighted sum of hi 
3.  is computed by 
4.  is computed by 

* **Multi-head Attention**

Input dimensions for keys = values = queries = dmodel.

1. Instead of performing a single attention function with dmodel - dimensional keys, values and queries. It’s better to linearly project the queries, keys and values h times with different, learned linear projections to dk, dk, and dv dimensions, respectively.
2. For these projected versions of queries, keys and values, we then perform attention function in parrel.
3. Output dv dimensional output values.
4. These above output values are concatenated and once again projected   

* **Position-wise Feed-Forward Network**

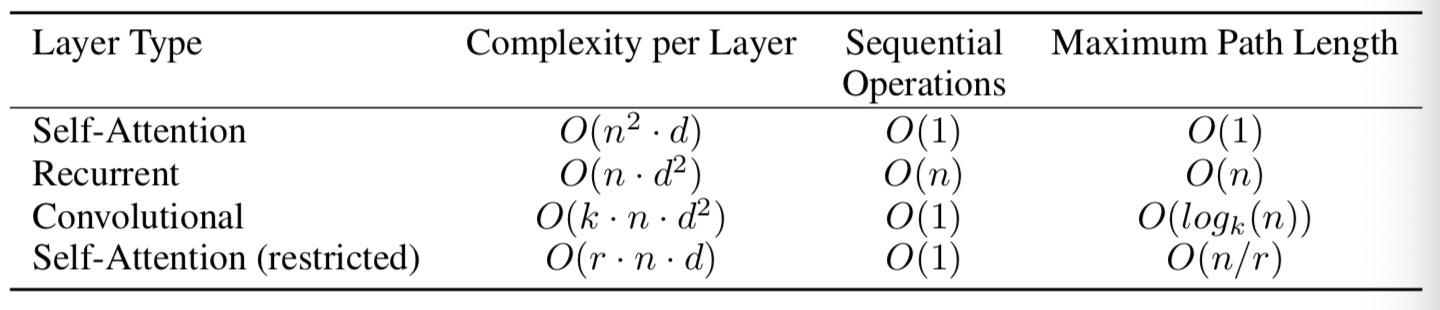
Fully connected feed-forward network, which is applied to each position separately and

Identically. This consists of two linear transformations with a ReLU activation in between.



Another way of describing this is as two CNNs with kernel size =1 . The dimensionality of input and output is dmodel = 512, and the inner-layer has dimensionality dff= 2048.

* **Why self-attention**
* Total computational complexity per layer.
* The amount of computation can be parallelized, as measured by the minimum number of sequential operations required.
* The ability to learn long-range dependencies.



# Summary about transformer

# Multi-head:

# Aim to learn more aspects meaning of each word (such as bank, one meaning is river bank, and another is financial relative bank). From insights, each head represents one aspect mean like river bank.

# And the embeddings for this head is representing some factors that can describe this aspect meaning. For example, for the bank meaning, one dimension in the embeddings may represent something for financial, and another dimension may mean someplace we can withdraw money, etc.

# For the Q, V, K and softmax

# Use linear with q, v, k to transfer the original hidden\_size to int (hidden\_size/num\_head)\*each\_head\_size.

# Aim to learning the contextual interaction for each word with the rest words in the sentence, using attention score to get the importance for each word to other rest words in the sentence.

# For example: In encoder, a sentence’s shape = (, len, num\_head, each\_head\_emb), the quey is the same (, len, num\_head, each\_head\_emb), and key would also be the same shape.

# Transpose for attn, key and query’s shape =(, num\_head, len, each\_head\_emb)

# Attn\_score = <quey, key> ‘s shape = (,num\_ head, len, len). ------- represent the contextual interaction importance for each word with the rest word.

# Softmax: sum the total attn\_score for all words in the sentence to represent the importance of this word for all sentence.

# 2，Bert

## What problems have been solved in this paper ?

## Model architecture

## Breakthrough components

## Training regime

## Tasks

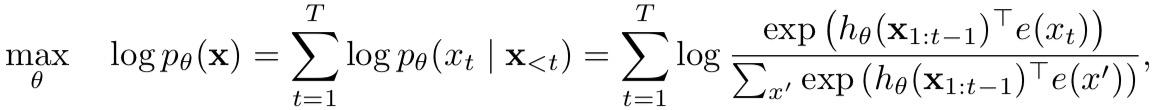
# 3，Xlnet

## What problems have been solved in this paper ?

Relying on corrupting the input with masks, BERT neglects dependency between the masked positions and suffers from a pre-train finetune discrepancy.

***Autoregressive(AR) language modeling***

* ***AR model such as RNNs or Transformers*** seeks to estimate the probability distribution of a text corpus with an autoregressive model. For instance, given a text sequence , AR model factorizes the likelihood into a forward product  or a backward one . A parametric model (e.g. a neural network) is trained to model each conditional distribution.
* ***AR model function:*** given a text sequence , AR model performs pretraining by maximizing the likelihood under the forward autoregressive factorization:

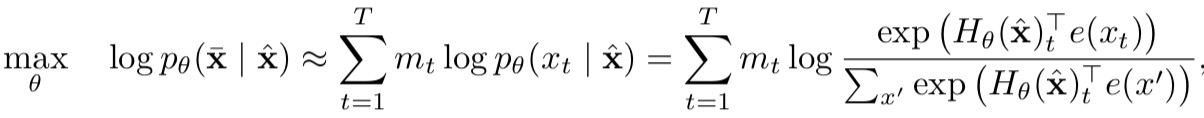
******

****** is a context representation produced by neural models, such as RNNs or Transformers.  denotes the embedding of x.

* ***Flaws of AR models:*** 1）AR model is only trained to encode a uni-directional context(either forward or backward), it’s not effective at modeling deep directional contexts. 2) As most downstream language understanding tasks often require bidirectional context information. This results in a gap between AR model and effective training.

***Autoencoding(AE) language modeling***

* ***AE model such as BERT*** does not perform explicit density estimation but instead aims to reconstruct the original data from corrupted input. For instance, BERT. Given the input token sequence, a certain portion of tokens are replaced by a special symbol [MASK], and the model is trained to recover the original tokens from the corrupted version. Since density estimation is not part of the objective, BERT is allowed to utilize bidirectional contexts for reconstruction. As an immediate benefit, this closes the aforementioned bidirectional information gap in AR model and lead to improved performance.
* ***AE model function:*** given a text sequence , BERT
* first constructs a corrupted version  by randomly setting a portion (e.g. 15%) of tokens in x to a special symbol [MASK]. Let the masked tokens be 
* the training objective is to reconstruct  from :



where  indicates is masked, and  is a Transformer that maps a length-T text sequence x into a sequence of hidden vectors 

* ***Flaws of AE models:***
* **Input noise:** The artificial symbols like [MASK] used by BERT during pretraining are absent from real data at finetuning time, resulting in a pretraining-finetuning discrepancy.
* **Independence Assumption:** As emphasized by thesign, BERT factorizes the joint conditional probability  based on an independence assumption that all masked tokens are separately reconstructed. In comparison, the AR model objective factorizesusing the product rule that holds universally without such an independence assumption.
* **Context dependency:** The AR representation  is only conditioned on the tokens up to position t (i.e. tokens to the left), while the BERT representation  has access to the contextual information on both sides. As a result, the BERT objective allows the model to be pretrained to better capture bidirectional context.

***XLNet : a generalized autoregressive method that leverages the best of both AR and AE while avoiding their limitations.***

* Instead of using a fixed forward or backward factorization order as in conventional AR models, XLNet maximizes the expected log likelihood of a sequence ***with all possible permutations of the factorization order:*** the context for each position can consist of tokens from both left and right. Therefore, each position learns to utilize contextual information from all positions and capture bidirectional context.
* As a generalized AR model, XLNet does not rely on data corruption and suffer from the pretrain-finetune discrepancy that BERT is subject to.
* The Autoregressive objective also provides a natural way to use the product rule for factorizing joint probability of predicted tokens, eliminating the independence assumption made in BERT.
* Improves architectural designs for pretraining: 1) XLNet integrates the segment recurrence mechanism and relative encoding scheme of Transformer-XL into pretraining, which empirically improve the performance especially for tasks involving a longer text sequence. 2) Naively applying a Transformer(-XL) architecture to permutation-based model does not work because the factorization order is arbitrary and the target is ambiguous. XLNet propose to reparameterize the Transformer(-XL) network to remove the ambiguity.

## Model architecture