Week 8 Notes

Transformer Networks 🤖

Purpose of encoder: Low entropy info to high entropy, lower dimension representation/encoding

Purpose of decoder: Making sense of the representation to produce something

Positional encoding:

- Using sin & cos:
    - to be able to be used with longer lengths than training examples
    - "We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions," (from the Attention paper)

Encoder-Decoder Attention similarities:

- Using decoder hidden states as queries, encoder attention scores as keys, encoder hidden states as "keys"

Multi-Head attention:

- *Self-attention* is achieved through passing Q, K, V from itself to itself
- At each sub-layer, there is normalisation between old Q, K, V and new. Therefore retaining dimensionality is needed.
- *Multi-headed*-ness: For every head, it just the same thing but with different weights. This allows each attention head to pay attention to different parts/representations.
- Why self-attention? Because this is how to encode the history up to the point - similar to the hidden states in an RNN.

> **Why residual connections?**
>
> - The result of multi-head attention is (h * d_h)-dimension after concatenation. So d_h is normally d_{model} / h to be able to perform the addition.
> - Then you "put them back" into the original sequence in some way.
> - LayerNorm needs to take place to prevent values from exploding

Feed-Forward Network

- Needed to "combine" the different inputs - allows more complex interactions between attention vectors

Decoder Layers:

- Only different from encoder with *masking*. This is to allow only attention to scores from the previous words in the sequence. This is only for training.
- Masking is only applied to self-attention sub-layer

CNNs

Why CNN?

- Faster than RNN (due to being more parallelized)
- CNN good at extracting signals

Convolution

- For NLP: using 1-d convolution
- Image recognition: 2-d convolution

Architecture in 2014 paper: *n* words * *k* vectors -> filter -> max-over-time pooling

Generating a feature map

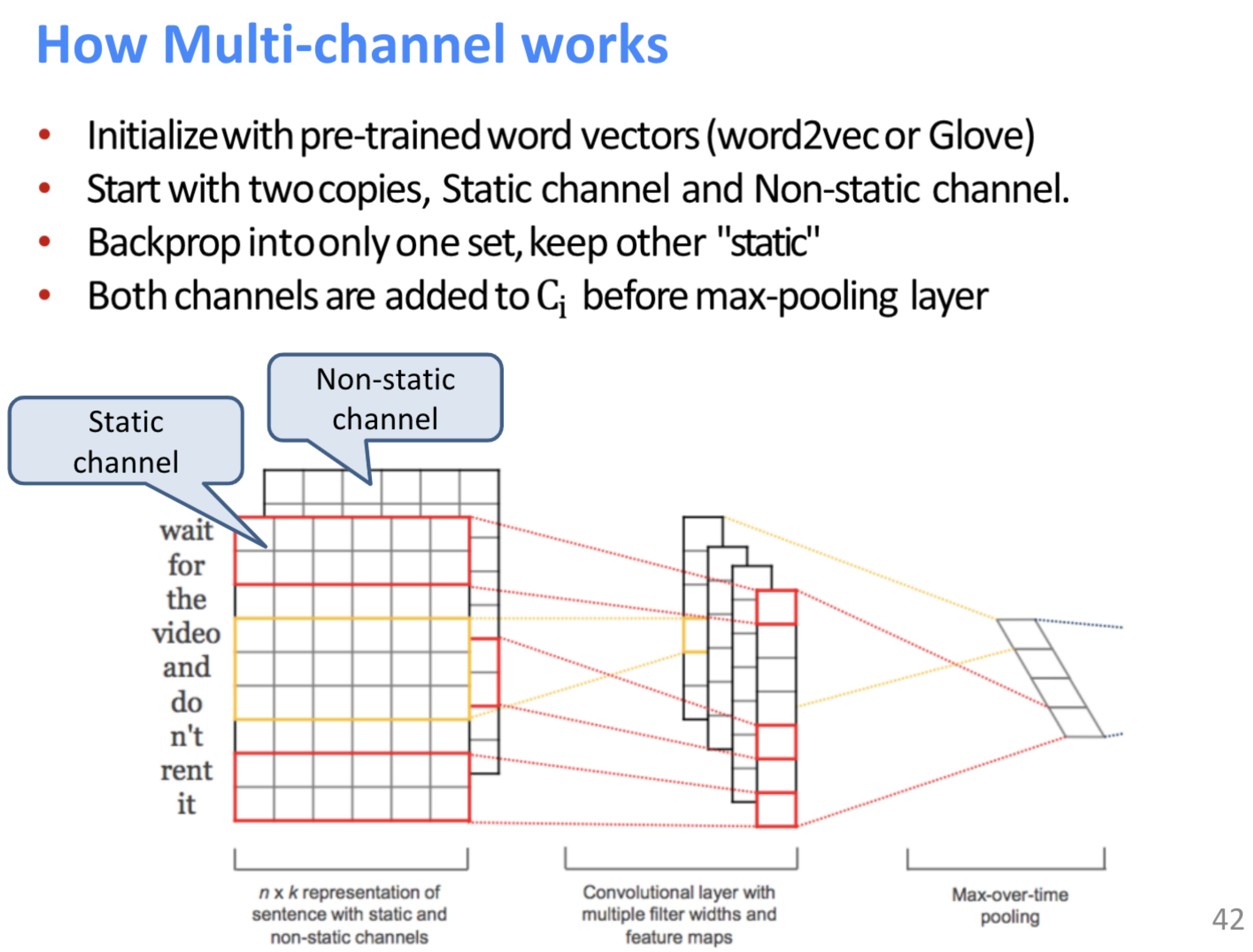
- Sentence with *n* words
- window of length *h*
- Result is a feature map in $\boldsymbol{R}^{n-h+1}$

Why do we take the maximum value (in pooling)?

- Reduce dimensionality
- Helps to capture the strongest signal

Multi-channel approach

- To prevent overfitting
- How to use:



Character-level CNN

Advantages: Model is robust against typos and misspelling, and can be used for strings like URLs

Disadvantages: Longer to train

Quantization

- Encoding target characters as one-hot vectors

Larger datasets tend to perform better in CLCNN.

> *"No Free Lunch Theorem"*

URL Example: Sum pooling used to "accumulate" rather than max-pooling

Papers Referenced: