Transfer Learning in NLP

Self-supervised learning tasks and model capacity in NLP

Snorre Ralund, Ph.D Fellow, University of Copenhagen, SoDaS

KØBENHAVNS UNIVERSITET

Transfer Learning in NLP

Self-supervised learning tasks and model capacity in NLP

Snorre Ralund, Ph.D Fellow, University of Copenhagen, SoDaS

KØBENHAVNS UNIVERSITET

Self-Supervised Learning

- Extract and use the metadata and relevant context as supervisory signal.

Self-Supervised Learning

- Extract and use the metadata and relevant context as supervisory signal.
- Self-referential prediction tasks.

E.g. Leverage large scale user generated tags

- Hashtags for expressing topic and summarization.
  - #sarcasm #irony
  - #topic
- Emojies to explicate emotional intention - DeepMoji

Self-Supervised Learning

- Extract and use the metadata and relevant context as supervisory signal.
- Self-referential prediction tasks.

E.g. Leverage large scale user generated tags

- Hashtags for expressing topic and summarization.
  - #sarcasm #irony
  - #topic
- Emojies to explicate emotional intention - DeepMoji

Other User generated tags

- Subreddits - including reactions
- Keywords in Scientific Articles
- Tags in stackoverflow

Self-Supervised Learning

Language models: Supervisory signal from raw text

- Predict next word / Char given previous word
- Predict word given context. (Cooperation and interactions)
  - Word2Vec - Context window.
  - BERT – Removing random words from larger (32) Context Window
- Predict word given previous context + Reverse (ELMO)
- Predict next / previous sentence
  - E.g. SkipThoughts, BERT.
- Denoising better than Language models (Raffel et. al 2019)

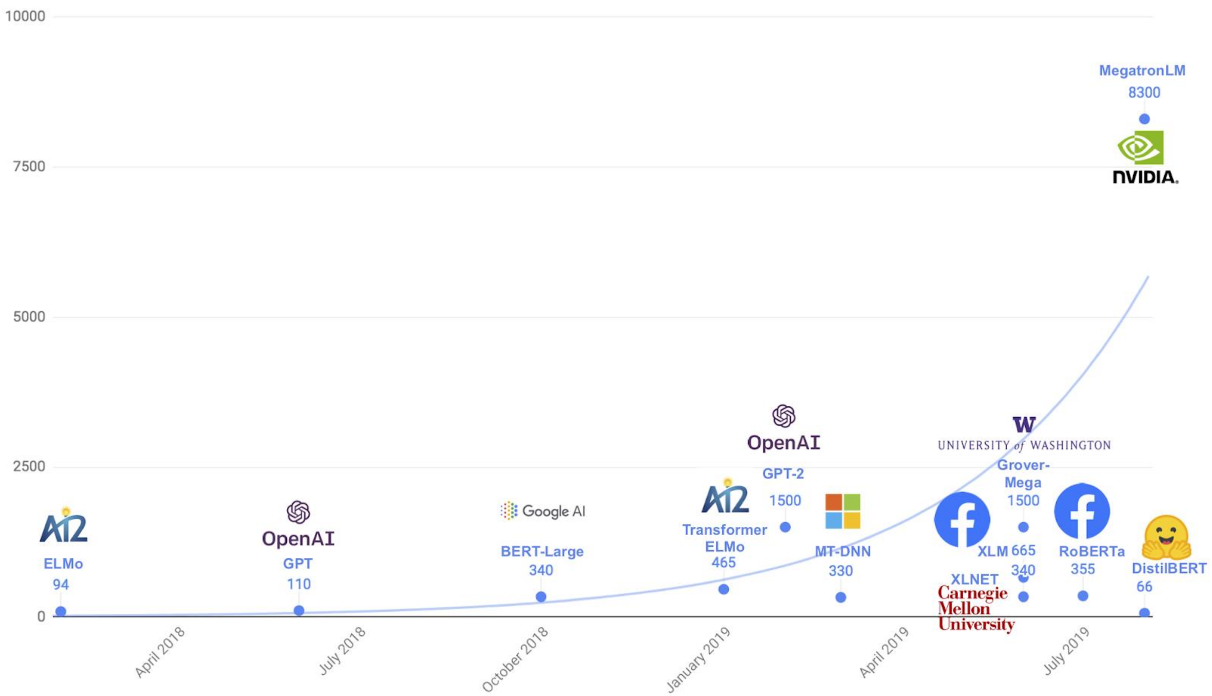
External Reference

- View the Director of One of the Major Transfer Learning Hubs, introduction to transfer learning.
  - https://www.youtube.com/watch?v=0T_Qr4qBrqc

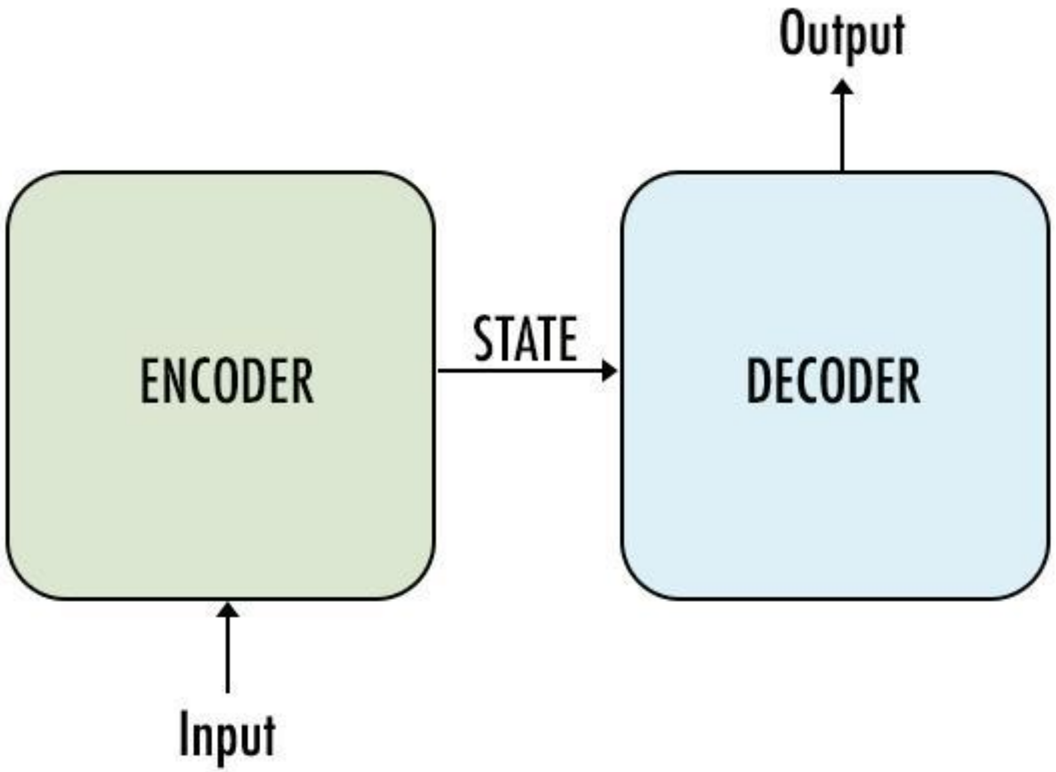

Data Science
fwdays
07.09.2019

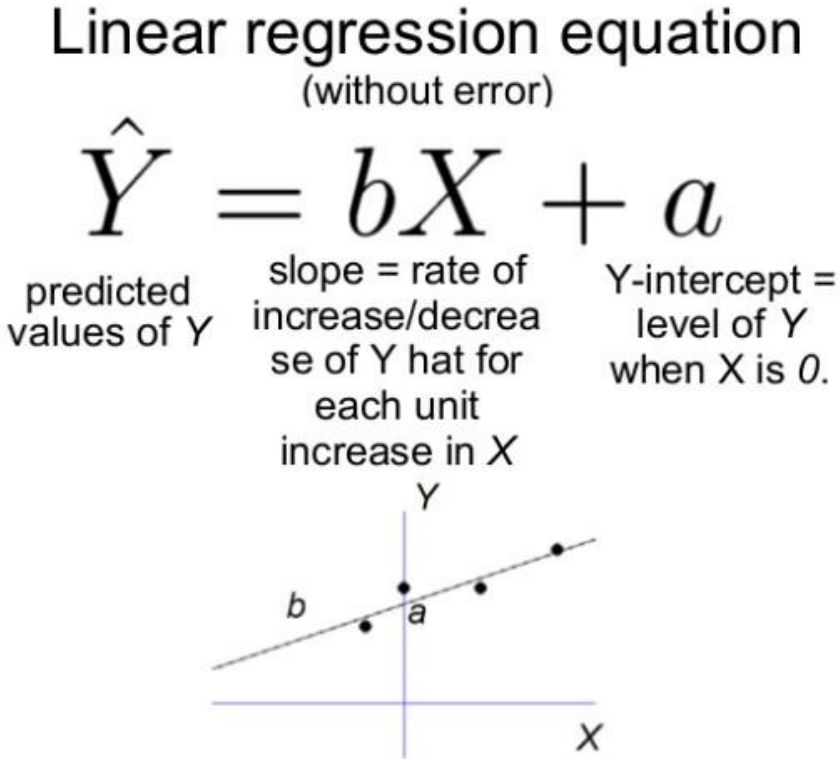
Reusing models

- Large Expensive Models can be re-used.
- Sharing via Hubs:
  - Transformers package
  - TFHUB
  - PYTORCH HUB
- Framework agnostic.



Model Capabilities: Input - Encode – Decode - Output





Model Capabilities

- Encode information from input into a Vector (or network of vectors)
  - Syntax, semantics, topical information, facts etc.
  - E.g.

    Dimensions: [Noun, Active, Animal]
    - Mouse -> [0.2,-0.3,1]
    - Cat -> [0.2, 0.3, 1]
    - Catch -> [-0.2,0.3,-0.5]

```
glove_200.most_similar(positive=['police','black'],negative=['white'])[0:5]

[('cops', 0.7516576051712036),
 ('officers', 0.6661906838417053),
 ('arrested', 0.6204742193222046),
 ('suspect', 0.6187559366226196),
 ('cop', 0.6156525015830994)]

glove_200.most_similar(positive=['police','white'],negative=['black'])[0:5]

[('cops', 0.7516659498214722),
 ('officers', 0.7105646133422852),
 ('authorities', 0.6782428026199341),
 ('arrest', 0.6773560047149658),
 ('officials', 0.662535548210144)]
```

- Decode information
  - Process the encoded information to produce output.

So which models have what capabilities

- Word2Vec, FastText only simple attenuation.
- 3 layers, and embeddings are averaged.
  - Learns linear information (similar to a BOW)

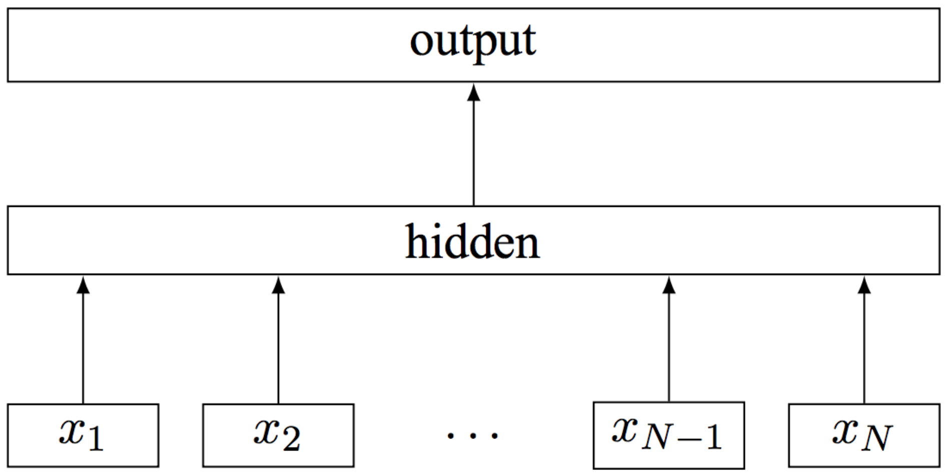
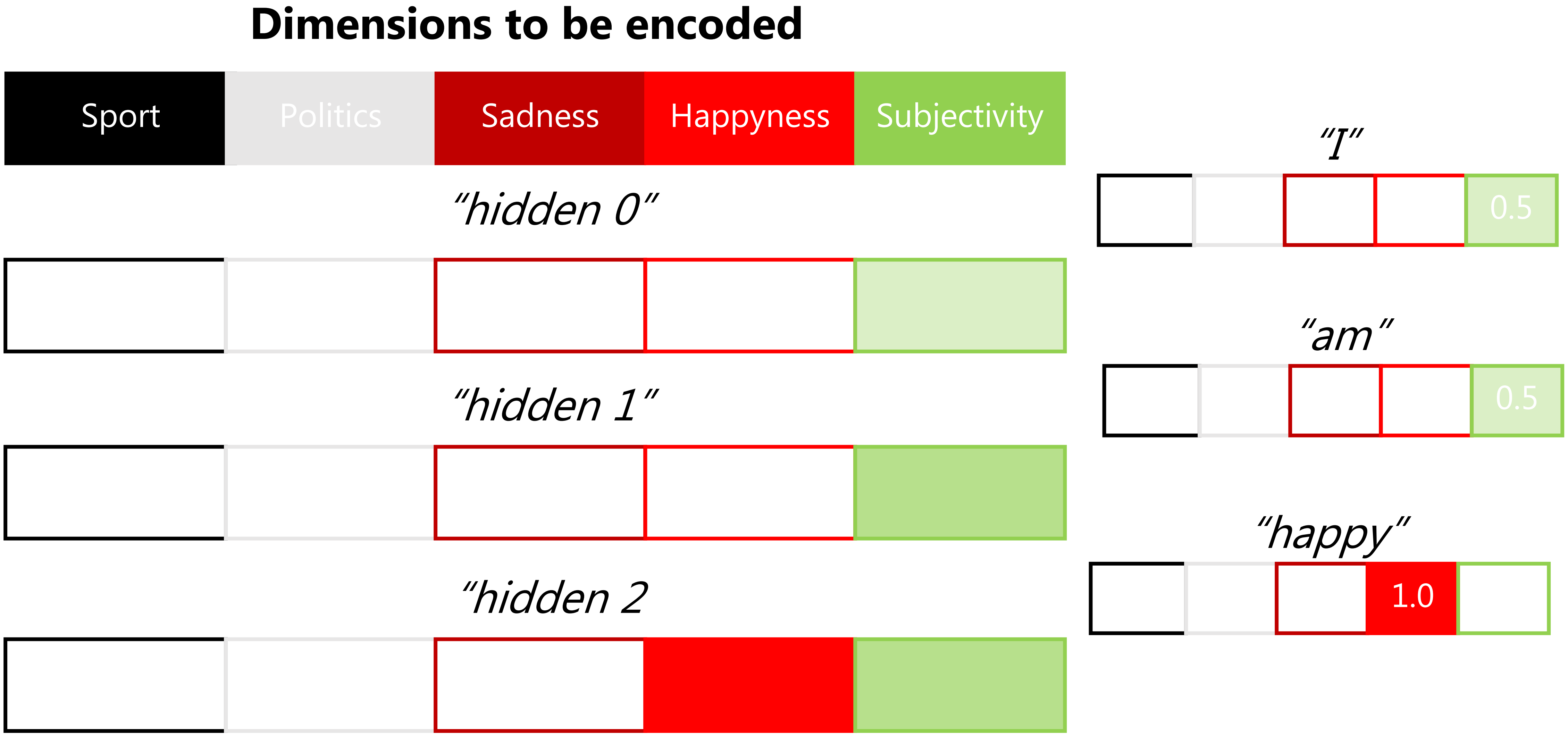


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \dots, x_N$. The features are embedded and averaged to form the hidden variable.

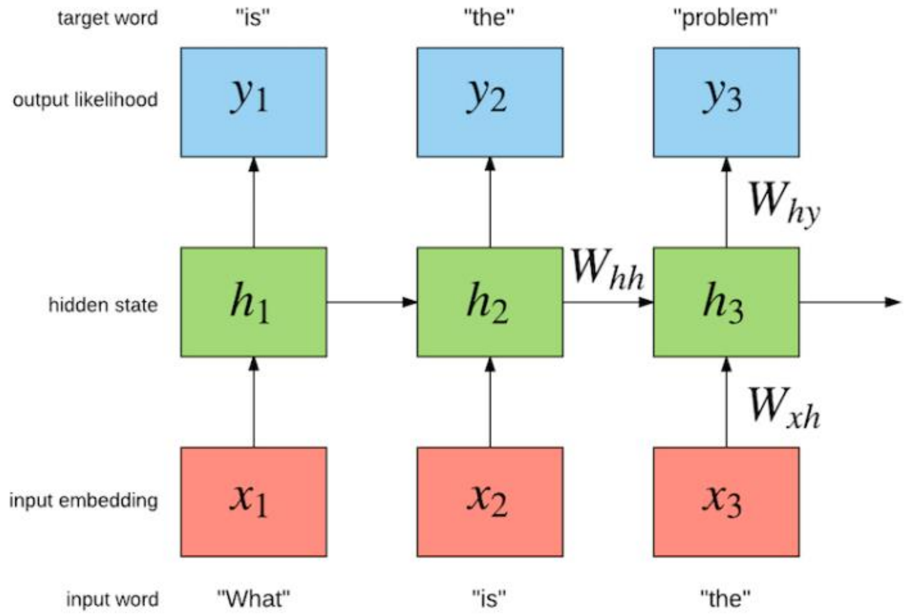
Simple Linear Transformation – Attenuation



So which models have what capabilities

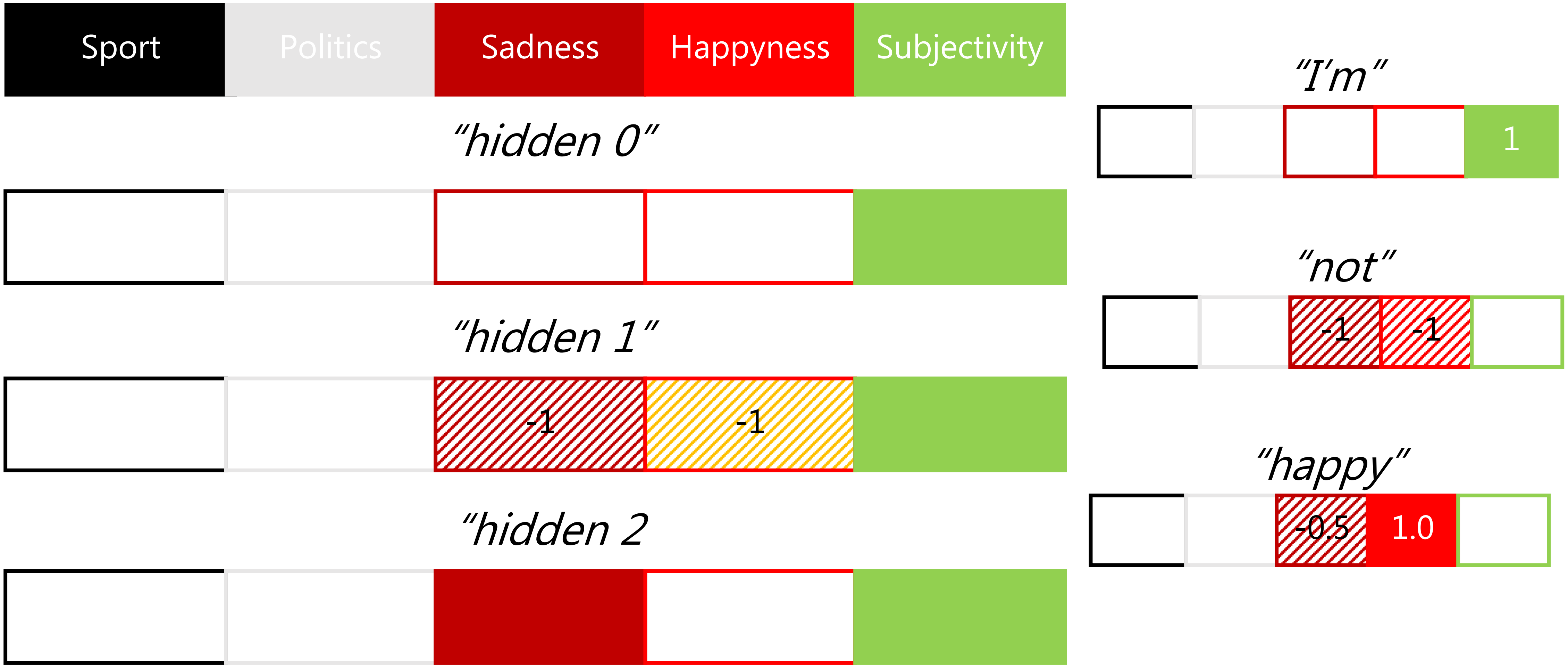
ELMo - Embeddings from Language Models. "Deep Contextualized word representations"

- Forward reading updating hidden states.
- Sequence of hidden states.

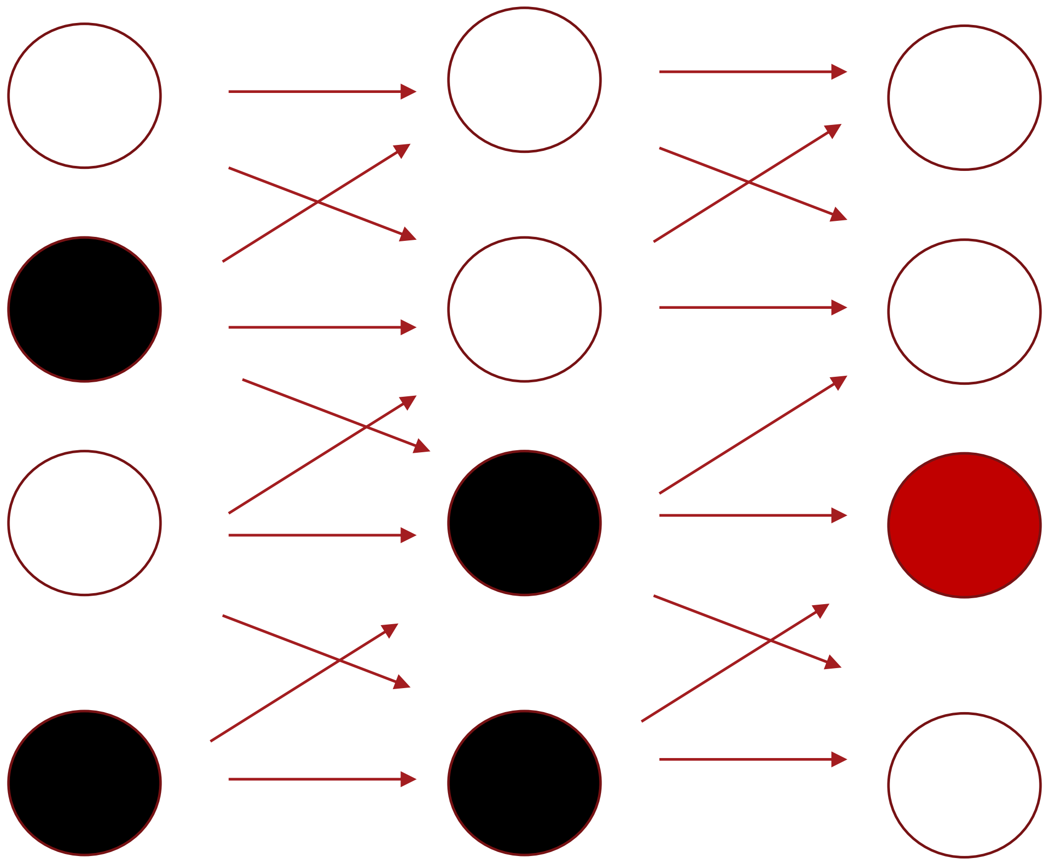


Simple Linear Transformation - Interaction

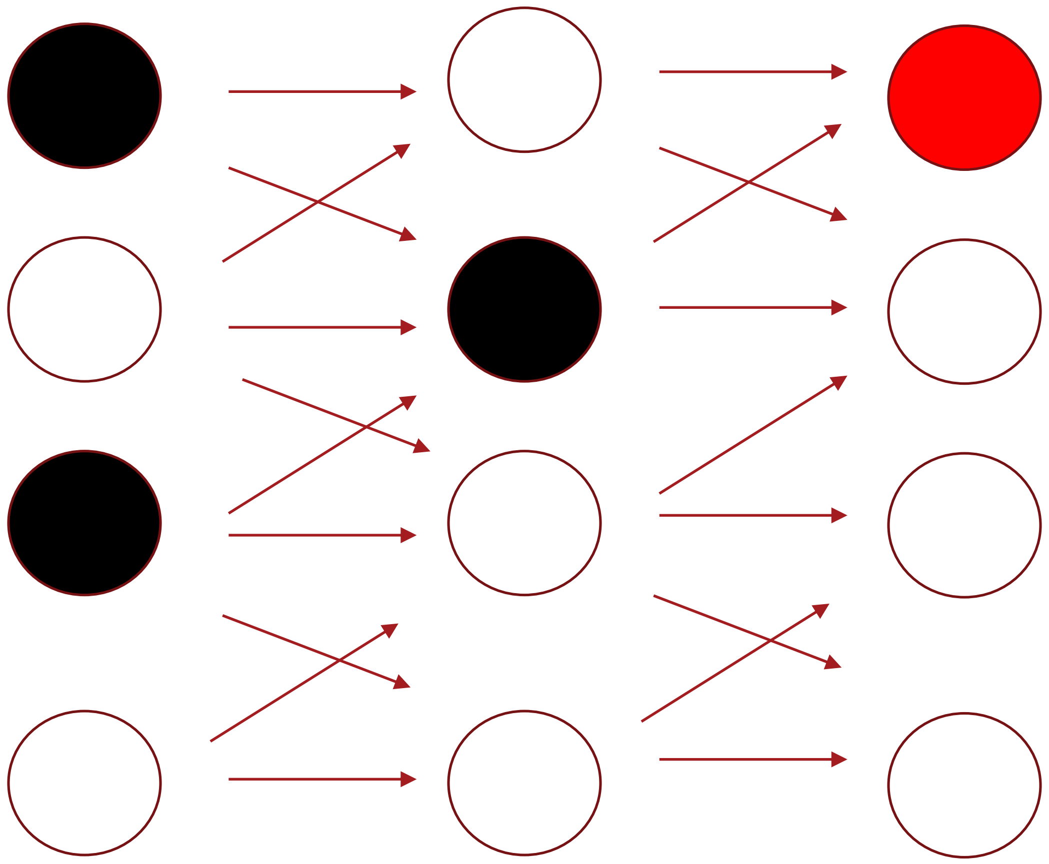
Dimensions to be encoded



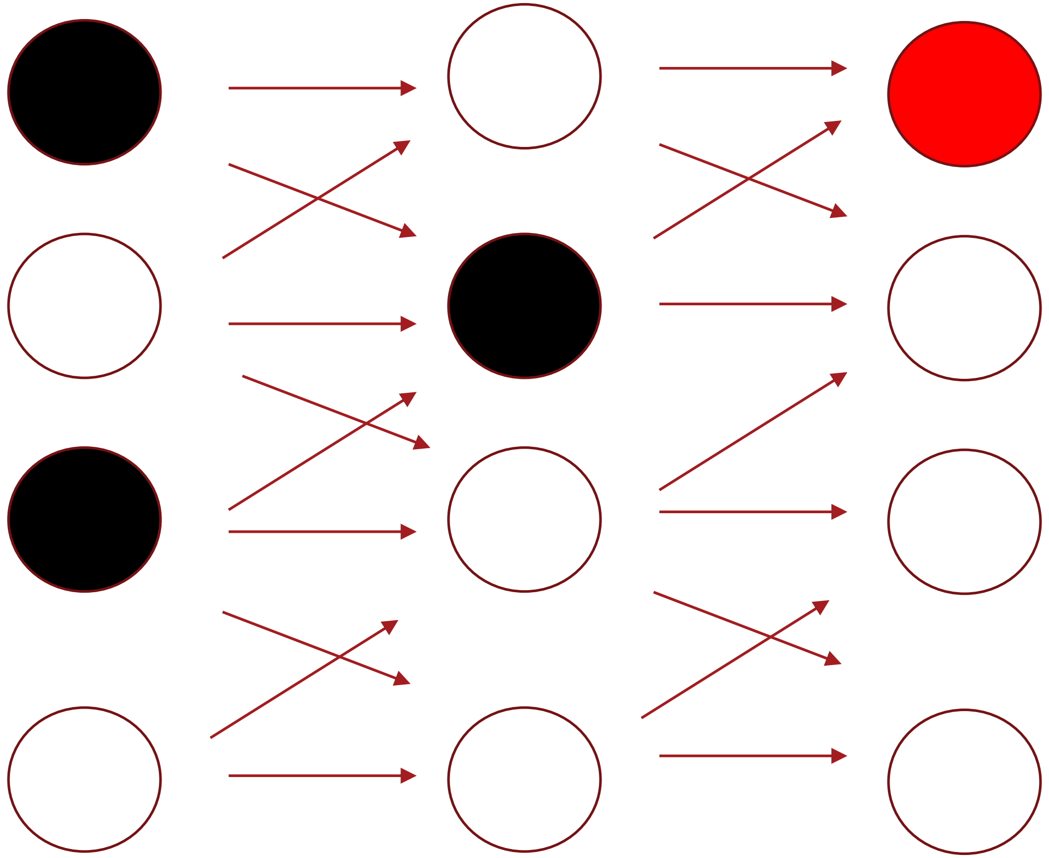
Non-linearities: Re-routing and Memory



Non-linearities: Re-routing and Memory



Non-linear transformation: Epistatic and Mutliple Interactions – Threshold



A by itself = NO Activation

B by itself = NO Activation

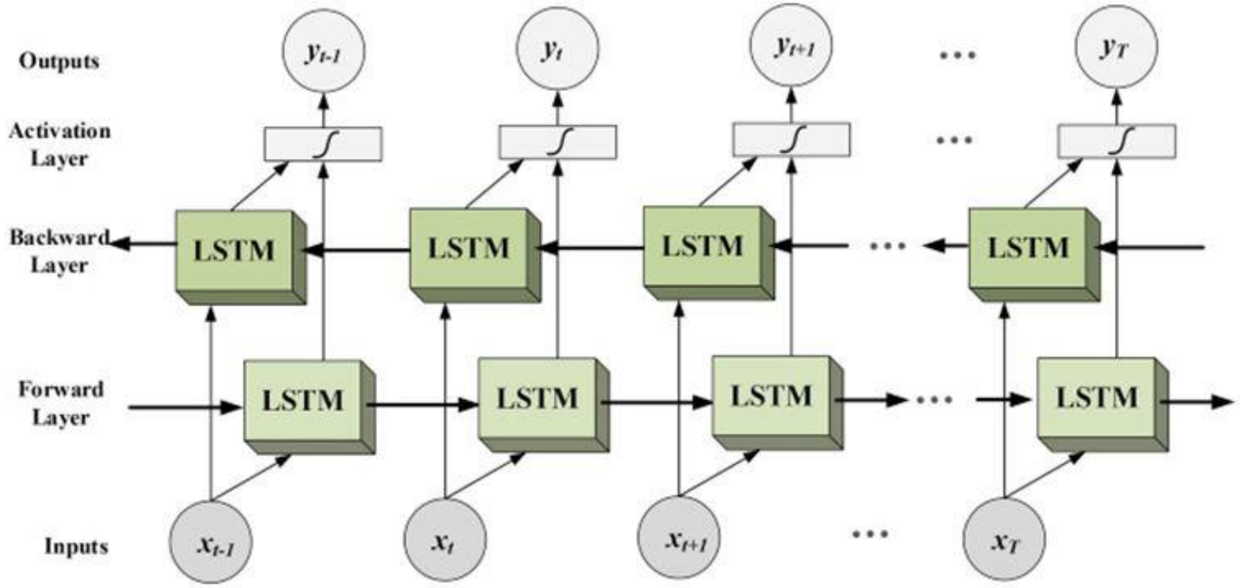
A+B = Activation

So which models have what capabilities

ELMo - Embeddings from Language Models. "Deep Contextualized word representations"

- Bi-directional: Forward and Backward.

- But only Sequential information



So which models have what capabilities

Bi-Directional models – Transformer Models

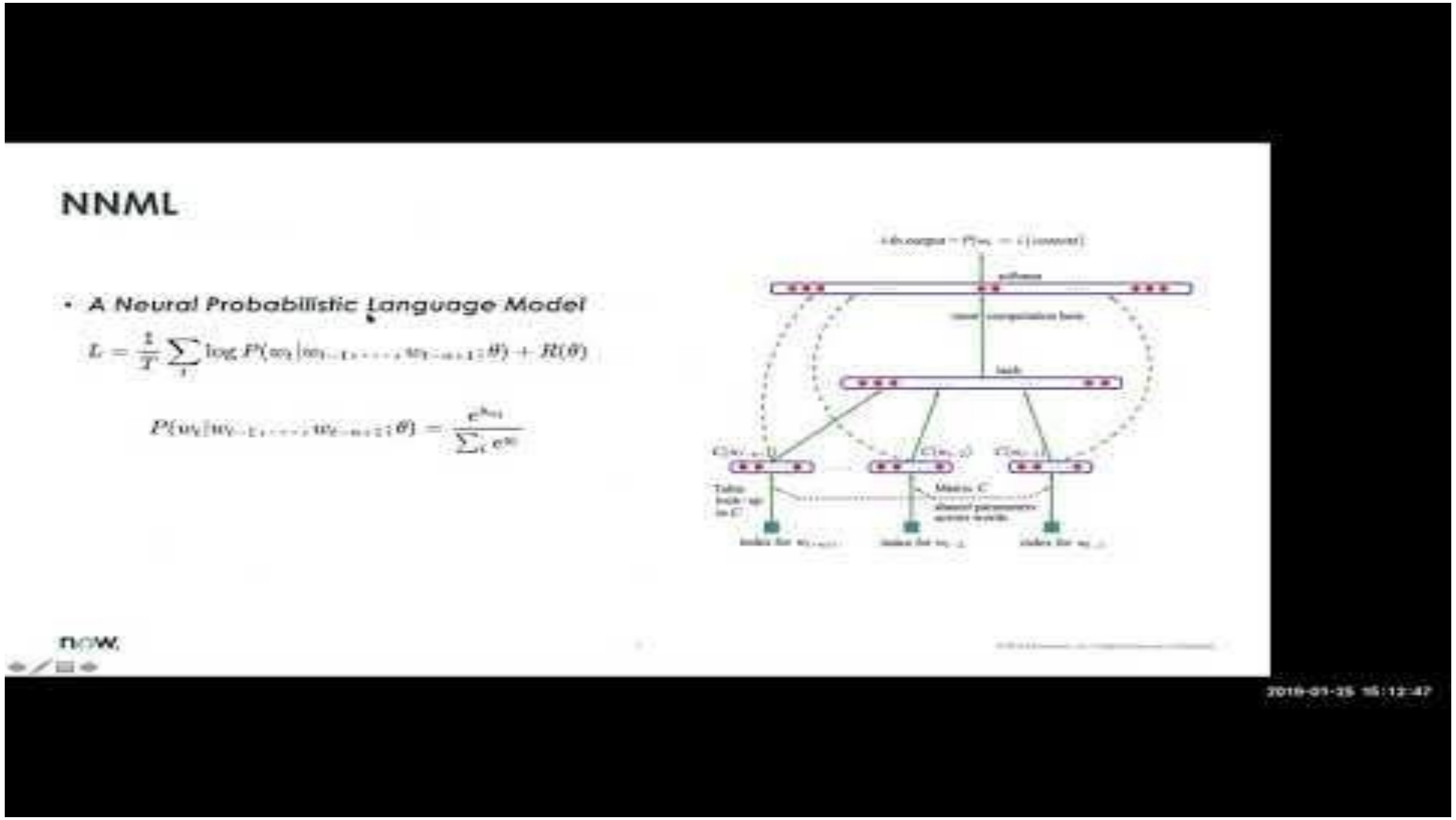
- BERT, GPT2, Electra ..
  - Allows word embeddings to interact by transforming each other.
  - Multiple transformation of each embedding is allow to interact multiple times, to allow for even more complex interactions.
    - See for a more detailed discussion: https://www.youtube.com/watch?v=ycXWAtm22-w

So which models have what capabilities

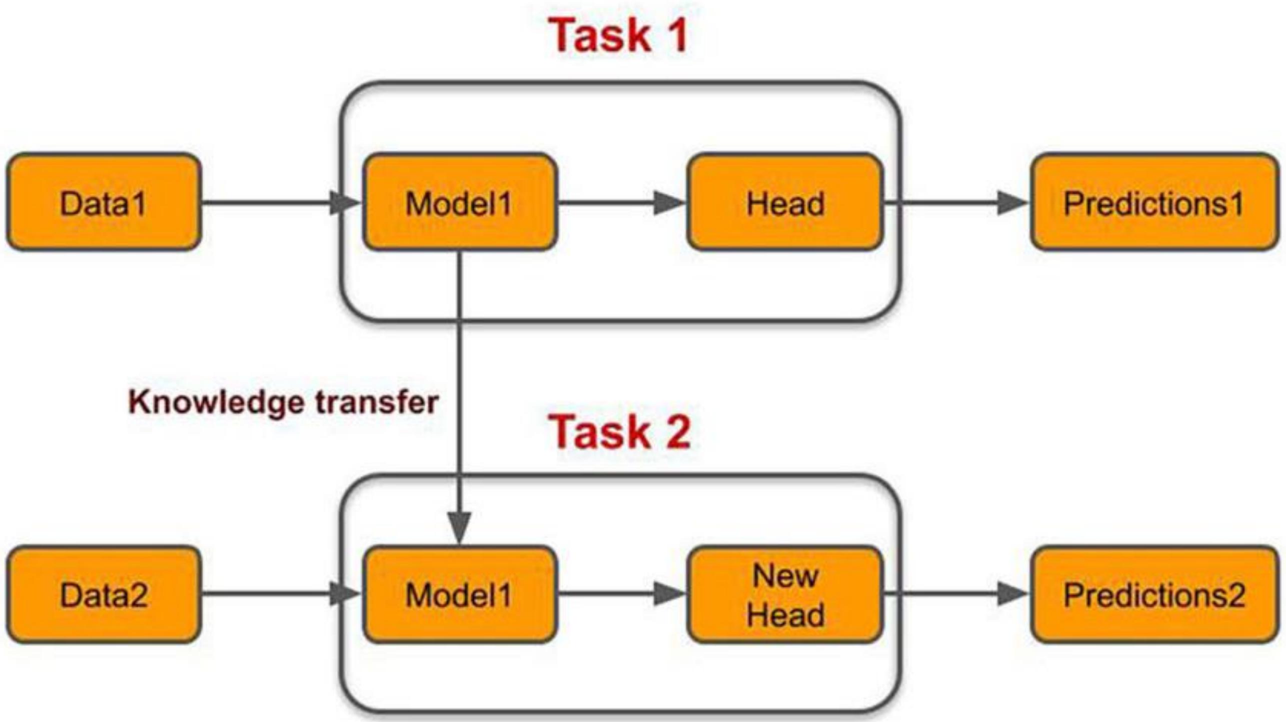
- Memory – Short and Long term.
  - Models struggle to incorporate long term knowledge in longer texts (Dailuk et al. 2017)
  - Open research question: Rae et. al 2019: "COMPRESSIVE TRANSFORMERS FOR LONG-RANGE SEQUENCE MODELLING"
  - BERT has a fixed context window.
  - XLNET, Transformer XL, Compressive Transformer are working in this direction.

External Reference 2

- View technical exposition of the history of language models
  - https://www.youtube.com/watch?v=ycXWAtm22-w


NNML
• A Neural Probabilistic Language Model
$L = \frac{1}{T}\sum_t \log P(w_t|w_{t-1},\ldots,w_{t-n+1};\theta) + R(\theta)$
$P(w_t|w_{t-1},\ldots,w_{t-n+1};\theta) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$

Transfer Learning: Adaptation



Transfer Learning: Adaptation

- Match Tokenization Scheme.

```
# Glove preprocessing (rewritten for python from the following ruby script:
#https://nlp.stanford.edu/projects/glove/preprocess-twitter.rb
import re
RE_URL = r'(?:https?://|www\.)(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'
RE_EMAIL = r'\b[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+(?= |$)'
RE_USER = r'(?:@\w+)(?= |$)'
#RE_HEART = r'(?:<+/?3+)+' # including broken heart.
RE_HEART = r'<3'
# Construct emoticons
eyes = r"[8:=;]"
noses = r"[\'`-]?"

mouth = r'[)dD]'
mouth_right = '[(]'
smiles = [eyes+noses+mouth,mouth_right+noses+eyes]
SMILE_RE = '|'.join(smiles)
LOL_RE = eyes+noses+'p'

frowns = [mouth+noses+eyes,
        eyes+noses+mouth_right]
SAD_RE = '|'.join(frowns)
neutral_mouth = r'[\/|l*]'
NEUTRAL_RE = eyes+noses+neutral_mouth
# Make sure it is not a hashtag.
ALLCAPS_RE = r'(?<!#)(?:\b|^)((?:[A-Z0-9_]*)?[A-ZÆØÅ]{2,}(?:[A-Z0-9_]*)?)(?:\b|$|[.!?])'
test = '#ASHDSH ASDKASLD990234 fFASF'

HASHTAG_ALLCAPS_RE = r'(#)(?:\b|^)((?:[A-Z0-9_]*)?[A-Z]{2,}(?:[A-Z0-9_]*)?)(?:\b|$|[.!?])'
HASHTAG_RE = r'(#)([a-zA-Z0-9_]+)'

regex_replace = [(RE_URL,'<URL>'),
                 (RE_URL2,'<URL>'),
                 (RE_USER,"<USER>"),
                 (RE_EMAIL,'<EMAIL>'),
                 ## ALLCAPS
                 (ALLCAPS_RE,r'<ALLCAPS> \1'),
                 #emoticons
                 (SMILE_RE,'<SMILE>'),
                 (LOL_RE,'<LOLFACE>'),
                 (SAD_RE,'<SADFACE>'),
                 (NEUTRAL_RE,"<NEUTRALFACE>"),
                 (RE_HEART,'<HEART>'),
                ## HASHTAGS
```

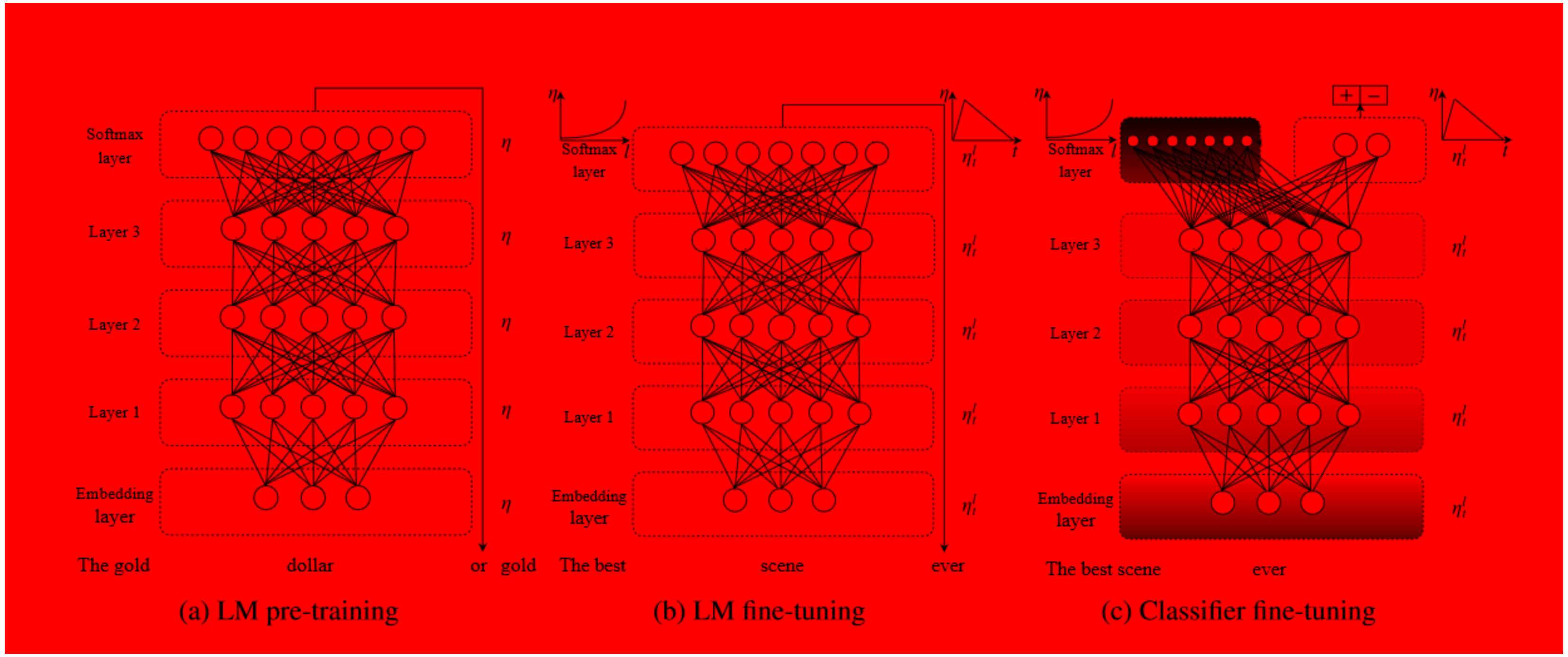
Transfer Learning: Adaptation

- Match Tokenization Scheme.

- Streamlining of this is developed around the package tokenizers:
  - https://github.com/huggingface/tokenizers

Transfer Learning: Adaptation

- Match Tokenization Scheme
- **Finetuning** of ***Language Model*** for domain adaptation. (Howard and Ruder 2017)



Transfer Learning: Adaptation

Finetuning for Classification

- Transfer techniques
  - Still an open research question, however.

Transfer Learning: Adaptation

Finetuning for Classification

Transfer techniques

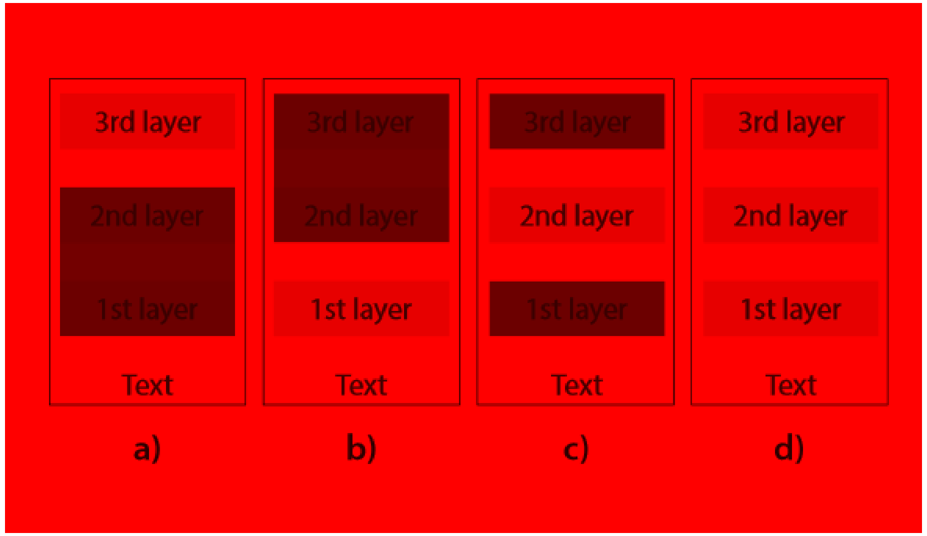
- How to Decode?
  - Use word embeddings directly for word level predictions:
    - E.g. parsing.
  - Summarize Embeddings to form Sentence level representation.

  - Classifier heads: Simple linear Softmax, or LSTM.

Transfer Learning: Adaptation

Transfer Techniques

- How To Train?
  - Avoid overfitting: large model adopt to small data -- memorization
  - Discriminative learning rates (earlier layers have more fundamental information → lower learning rate → reduce updates.)
  - Gradual unfreezing of layers (Howard and Ruder 2017), Chain-thaw (Felbo et al. 2017)



Transfer Learning: Adaptation

Transfer Techniques

- How To Train?
  - Avoid overfitting: large model adopt to small data -- memorization
  - Discriminative learning rates (earlier layers have more fundamental information → lower learning rate → reduce updates.)
  - Gradual unfreezing of layers (Howard and Ruder 2017), Chain-thaw (Felbo et al. 2017)
  - Auxillary loss function (keeping the Language model objective) (Chronopoulou et. al.2019)