Lecture 14: Transformers and Self-Attention

Ashish Vaswani and Anna Huang

Learning Representation of Variable Length Data

RNNs

- LSTMs, GRUs and variants dominate recurrent models
- Natural fit for sentences and sequences of pixel

But

Sequential computation inhibits parallelization

No explicit modeling of long and short range dependencies

Attention ⇒ why not use attention for representation

Text generation

Classification & regression with self attention

[Parikh et al. 2016, Lin et al. 2016]

Self attention with RNNs [Long, et al. 2016, Shao, Gows et al. 2017]

Recurrent attention [Sukhbaatar et. al. 2015]

Attention is cheap

[FLOPS]

Self Attention $O(length^2 \cdot dim)$ if length $\leq$ dim

RNN(LSTM) $O(length \cdot dim^2)$

Convolution $O(length \cdot dim^2 \cdot kernel\_width)$

Multi-head attention:

this is to put attention in different positions seperately

Importance of Residuals

Residuals carry positional information to higher layer, among other information

Music Generation

finding similar motifs

Probabilistic Image Generation

Texture synthesis with self similarity

Non-local NN (Wang 2015)

Selfattention

$$A(Q,K,V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)V$$

Image Transformer

Combining Locality with self Attention

Image Transformer - ICML 2018

Music generation using self Attention

Music Transformer (ICLR 2019)

→ score → performance → sound →

composer performon instrument listener

$$\text{Attention} = \text{softmax}\left(QK^T + \text{skew}\left(QE_{rel}^T\right)\right)$$

~~Evivalence~~

Relative positions. Translational Equivalence

- Fast Decoding sequence models ICML 2018 Kaiser
- Better understanding of Vector Quantized autoencoders Roy 2018
- Blockwise Parallel Decoding for Autoregressive model (NeurIPS, 2019) Stern

Lecture 15: Natural Language Generation (NLG)

Neural approaches to NLG

Section 1: LM and decoding algos; NLG subcomponent

- Machine Translation
- (Abstractive) summarization
- Dialogue (chit-chat and task-based)

- Creative writing: storytelling, poetry generation
- freeform QA (i.e. answer is generated)
- Image Captioning

Language Modeling: The task of predicting the next word, given the words so far

$$P(y_{t+1} | y_1, y_2 \cdots y_t)$$

- A system that produces this probability distribution is called a Language model

Conditional Language model:

$$P(y_t | y_1, y_2, \cdots y_{t-1} | x)$$

Decoding algo

- Greedy Decoding
- Beam Search
  - ⇒ search algorithm which aims to find a high probability sequence (non-necessary the optimal sequence, though) by tracking multiple possible sequences at once

What's the effect of changing beam size k?

- Small k has similar problems to greedy decoding
  → ungrammatical, unnatural, nonsensical, incorrect
- Larger k means you consider more hypotheses
  → Increasing k reduces some of the problem above
  → Larger k is more computationally expensive
  → Increasing k too much decreases BLEU score. This is primarily because large-k beam search produces too short translations
  → In open ended tasks (large-k can make output more generic

Sampling-based decoding

- Pure sampling
  → On each step t, randomly sample from the prob dist P_t to obtain your next word
  → Like greedy decoding but sample based insted of argmax

Top-n sampling:

→ On each step t, randomly sample from P_t restricted to just the top-n most probable words.

→ Increasing n to get more diverse/risky output

→ Decrease n to ge more generic/safe output

both of them are more efficient than beam search

Softmax temperature:

On timestep t, the LM computes a prob dist P_t by applying the softmax function to a vector of scores $s \in \mathbb{R}^{|V|}$

$$P_t(w) = \frac{\exp(s_w)}{\sum_{w' \in V} \exp(s_{w'})}$$

we can apply a temperature hyperparameter τ to the softmax

$$P_t(w) = \frac{\exp(s_w/\tau)}{\sum_{w' \in V} \exp(s_{w'}/\tau)}$$

→ Raise the temperature τ: P_t becomes more uniform

. Thus more diverse output (probability is spread around vocab)

→ Lower the temperature τ: P_t becomes more spiky

. Thus less diverse output

Softmax temperature is not a decoding algo

We can apply this during test time, in conjunction with a decoding algorithm (such as beam search or sampling)

Section 2: NLG tasks and neural approaches to them:

Summarization

Gigaword ⇒ news article → headline

LCSTS ⇒ paragraph → sentence summary

NYT, CNN/DailyMail ⇒ news article → (multi) sentence summary

Wikihow: full how-to article → summary sentences

Sem:

: Sentence simplification is a different by related task

→ rewrite the source text in a simpler way

• Simple Wikipedia : standard wiki/idea sentence → simple version

• Newsela : news article → version for children

Two main strategies:

Extractive summarization

Select parts (typically sentences) of the original text to form a summary

• Easier

• Restrictive (no paraphasing)

Abstractive Summarization

Generate new test using natural ~~text~~ LG techniques

• More difficult

• More flexible (more human)

Evaluation:

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

~~Rouge-N~~:

like BLEU, it's based on n-gram overlap

- ROUGE has no brevity penalty
- ROUGE based on recall, while BLEU is based on precision

* Precision is more important for MT and recall is more important for summarization (assuming you have a max length constraint)
* However, often a f1 version of ROUGE is reported anyway
- BLEU is reported as a single number, which is combination of the precisions for n=1, 2, 3, 4 n-grams
- ROUGE scores are reported seperately for each n-grams

ROUGE-1 → unigram overlap

ROUGE-2 → bigram overlap

ROUGE-L → LCS overlap

Neural approaches for summarization (2015-present)

→ 2015: Rushel al. publish the first seq2seq summarization paper

→ Single-document abstractive summarization is a translation task!

Thus we can apply standard seq2seq + attention NMT methods

Since 2015 → developments

- Making it easier to copy [prevent too much of it]
- Hierarchial / multi-level attention
- More global / high-level content selection
- Using RL to directly maximize ROUGE or discrete goals
- Resurecting pre-neural ideas (graph algos) and

working thms ~~un~~ into neural systems

Copy Mechanism:

- Seq2seq + attentions are good at writing fluent output but bad at copying over details correctly
- Copy mechanism: use attention to enable seq2seq system to easily copy words and phrases from the input to the output.
  - ⇒ Allowing both copying and generating gives us hybrid extractive/abstractive approach

$$P(w) = P_{gen} P_{vocab}(w) + (1 - P_{gen}) \sum_{i: w_i = w} a_i^t$$

- Big problems with copying mechanisms
  - They copy too much
  - bad at overall content selection

Preneural summarization had seperate stages for content selection and surface realizatin

Bottom up summarizatin:

→ Content selection stage: Use a neural sequence-tagging model to tag words as include or don't include

→ Bottom-up attention stage: The seq2seq+attention system can't attend to words tagged don't-include (apply a mask)

Simple but effective →
- better overall content selection
- Less copying

Neural summarization via RL:

Use RL to directly optimize ROUGE-L

ML+RL gives best result

Dialogue

- Task-oriented dialogue
  - → Assistive (customer service, recommendations)
  - → Co-operative (two agent solve a task together)
  - → Adversarial (two agents compete in a task through dialogue)
- Social dialogue
  - → chit-chat dialogue
  - → Therapy/mental wellbeing

Preneural dialogue systems used predefined templates or retrieve an appropriate response from a corpus of responses.

After 2015: open ended freeform dialogue.

→ Seq2Seq-based dialogue

However it quickly became apparent that a naive application of seq2seq has serious pervasive deficiencies for (chitchat) dialogue

- Genericness/boring responses
- Irrelevant responses (not
- Repetition
- Lack of context
- Lack of consistent persona

Irrelevant response problem:

Reason:
→ because it's generic
→ or because changing the subject to something unrelated

One solution: optimize for MMI (Maximum Mutual Information) between input S and response T

$$\log \frac{p(S,T)}{p(S)p(T)}$$

$$\hat{T} = \arg\max_{T} \{\log p(T/S) - \log p(T)\}$$

Genericness or boring response problem

→ Easy test-time fixes:

- Directly upweight rare words during beam search
- Use a sampling decoding algo rather than beam search

→ Conditioning fixes

- Condition the decoder on some additional content (e.g. sample some content words and attend ~~them~~ to them)
- Train a retrieve-and-refine model rather than a generate-from-scratch model.

Repetition Problems

Simple solution

→ Directly block repeating n-grams during beam search

More complex solutions

→ Train a coverage mechanism. - in seq2seq, this is an objective that prevents the attention mechanism from attending to the same words multiple times.

→ Define a training objective to discourage repetition

- If this is a non-differentiable function of the generated output, then will need some techniques e.g. RL to train

NLG: Storytelling:

Most neural storytelling work uses some kind of prompt:

- Generate a story-like paragraph given an image
- Generate a story given a brief writing prompt
- Generate the next sentence of a story given the story so far (story continuation)

Question: How to get around the lack of parallel data?

Answer: Use a common sentence encoding space

→ skip-thought vectors are a type of general-purpose sentence embedding method

- The idea is similar to how we learn an embedding for a word by trying to predict the words around it.

- Using COCO (an image captioning dataset), learn a mapping from images to the skip-thought encodings of their captions
- Using the target style corpus, & train a RNN-LM to decode a skip-thought vector to the original text.
- Put the two together

Generating story from a writing prompt

- In 2018, Fan et. al. released a new story generation dataset collected from Reddit's WritingPrompts subreddits
- Each story has an associated brief writing prompt
- propose a convolution based story model

* Gated Multihead multiscale self attention
  - The self-attention is important for capturing long-range context
  - The gates allow the attention mechanisms to be more selective
  - The different attention heads attend at different scales - this means there are different attention mechanisms dedicated to retrieving fine-grained information and coarse-grained information

Model-fusion:

- Pretain one seq2seq model, then train a second seq2seq model that has access to the hidden states of the first.
- The idea is that the first seq2seq model learns general LM and the second learns to condition on the prompt.

- The results are impressive
  - Related to prompt
  - Diverse; non-generic
  - Stylistically dramatic

However:

- Mostly atmospheric/descriptive/scene-setting; less events/plot.
- When generating for longer, mostly stays on the same idea without moving forward to new ideas - coherence issues.

NLG Evaluation:

Word overlap based metrics (BLEU, ROUGE, METEOR, F1, etc.)

- We know they are not ideal for machine translation worse for summarization and even worse for dialogue and storytelling

What about perplexity?

- Captures how powerful your LM is, but doesn't tell you anything about generation (e.g. if your decoding algo is bad, perplexity is unaffected)

Word embedding based metrics?

→ Compare the similarity of the word embeddings not just the overlap of the words themselves. Captures semantics in a more flexible way

→ still doesn't correlate well with human judgements for open ended task

Automatic evaluations of NLG

- We can define more focused automatic capture particular aspects of generated text
- Fluency (compute probability wrt a well trained LM)
- Correct style (prob w.r.t ~~well tr~~ LM trained on target corpus)
- Diversity (rare word usage, uniqueness of n-grams)
- Relevance to input (semantic similarity measures)
- Simple things like lengths and repetations
- Task specific metric e.g. compression rate for summarization

Though these doesn't measure overall quality, they can help us track some important qualities that we care about

Human evaluation:

- regarded as gold standard but slow and expensive
- does human evaluation solve all your problems? No!

Conducting human evaluation effectively is very difficult

Humans are inconsistent

- can be illogical
- lose concentration
- misinterpret your question
- can't always explain why they feel the way they do.

Adversarial discrimination

- Test whether the NLG system can fool a discriminator which is trained to distinguish human text from artificially generated text

Section 4:

Thoughts on NLG research, current trends and the future:

- Incorporating discrete latent variables into NLG
- Alternatives to strict left-to-right generation
- Alternatives to maximum likelihood training with teacher forcing

* NLG is the wildest part remaining in NN
- Neural NLG community is rapidly ~~exp~~ maturing
- Biggest roadblock is still evaluation

8 things on NLG

1) The more open-ended the task, the harder everything becomes

2) Aiming for specific improvement can also be more manageable than aiming to improve overall generation quality

3) If you're using a LM for NLG, improving the LM (i.e. perplexity) will most likely improve generation quality.

4) Look at your output a lot

5) You need an automatic metric, even if it's imperfect (probably several)

6) If we do human eval, make the question as focused as possible

7) Reproducibility is a huge problem in today's NLP + DL, and a "huger" problem in NLG

8) publicly release all your generated outputs.

8) Working with NLG is very frustrating.
But also very funny.

Non-autoregressive generation for NMT

In 2018 Gou et al published ~~Non-aut~~

- It doesn't generate the translation left-to-right, with each word depending on the ones before
- It generated the translation in parallel.

→ It has obvious efficiency advantages but it is also intriguing from NLP point of view

- Transformer based architecture and decoder can run in parallel at text time

Lecture 16: Coreference Resolution:

- Identify all mentions that refers to the same real world entity

  * NLP models cannot handle this properly.

Applications

- Full text understanding
  - 'information extraction', question answering summarization, ...
  - "He was born in 1961" (Who?)
- Machine translation
  - languages have different features for gender, number, dropped pronouns, etc.
- Dialogue systems.

Coreference Resolution:

1) Detect the mentions (easy)
2) Cluster the mentions (hard)

Mention Detection:

- Pronouns ... (POS tagger)
- Named entities (NER system)
- Noun phrases (constituency parsing)

Mention Detection: Not so simple

- Marking all pronouns, named entities and NPs as mentions

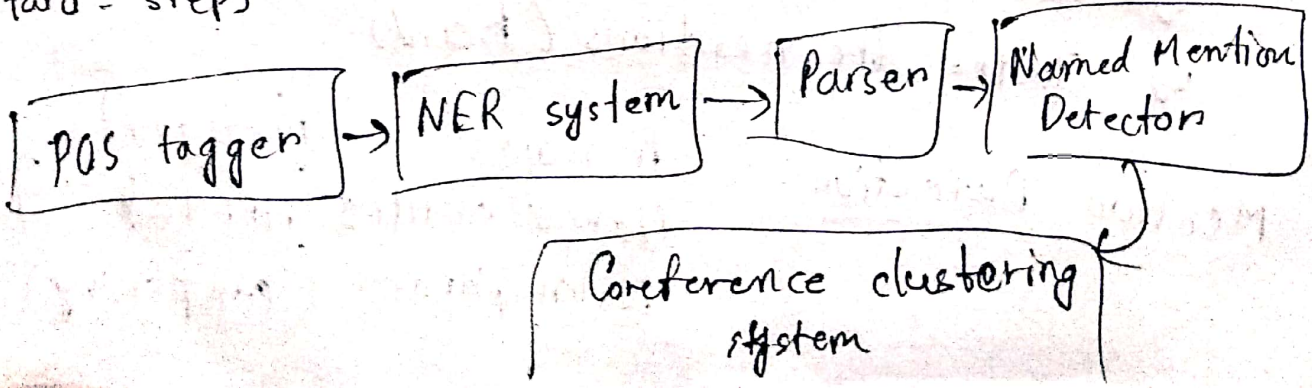
Are these mention?

- It is sunny
- Every student
- No student
- No student
- The best donut in the world
- 100 miles

How to deal with these bad mentions?

Can we avoid a pipelined system?

- We could instead train a classifier specifically for mention detection instead of using POS tagger, NER system, and parser.
- Or even jointly do mention-detection and coreference resolution end-to-end instead of two-steps



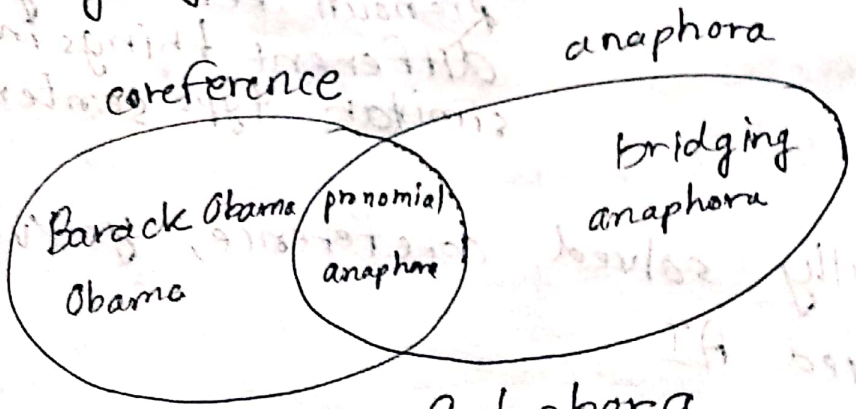
On to Coreference! First some linguistics

- Coreference is when two mentions refer to the same entity in the world.

→ A related linguistic concept is anaphora: when a term (anaphor) refers to another term (antecedent)

- The interpretation of the anaphor is in some way determined by the interpretation of the antecedent.

→ Not all anaphoric relations are coreferential bridging anaphora



Anaphora vs. Cataphora

- Usually the antecedent comes before the anaphor

Four Kinds of Coreference Models

- Rule-based (pronominal anaphora resolution).
- Mention Pair
- Mention Ranking
- Clustering

- Traditional pronominal anaphora resolution:
  Hobbs' naive algorithm (1976)
- Knowledge based Pronominal Coreference
  Winograd (1972) Winograd Schema
  ⇓
  pronoun referencing different things in similar type sentences

- If you've fully solved coreference, you've arguably solved AI

Mentim Pair

Train a Binary classifier that assigns every pair of mentions a probability of being coreferent

Mention Pair Models : Disadvantages

- Many mentions only have one clear antecedent
  - But we are asking the model to predict all of them

Solution: instead train the model to predict only one antecedent for each mention
- More linguistically plausible

Coreference Models: Mention Ranking:

- Assign each mention its highest scoring candidate antecedent

→ Dummy NA mention allows model to decline linking the current mention to anything

Positive examples: model has to assign a high probability to either one (but not necessarily both)

→ Solve → Apply a softmax over the scores for candidate antecedents so ~~probability~~ probabilities sum to 1.

Training:

- We want the current mention m_j to be linked to any one of the candidate antecedents it's coreferent with.
- Mathematically, we want to maximize this probability

$$\sum_{j=1}^{i-1} 1(y_{ij}=1) p(m_j, m_i)$$

- iterate throught candidate antecedents (previously occurring mentions)
- for ones that are coreferent to m_j
- we want the model to assign high probability.

- The model produce 0.9 probability for one of the correct antecedents and low probability for large everything else, and the sum will ~~be far~~ still be.

loss function

$$J = \sum_{i=2}^{N} -\log \left(\sum_{j=1}^{i-1} 1(y_{ij}=1) p(m_j, m_i) \right)$$

↑ usual trick of taking negative log to go from likelihood to loss.

Computing the probabilities

a) Non-neural statistical classifier (Extract feature and use

b) Neural Coref Model
- (word embedding and few categorical features)

c) End to end Model

SoTA [Lee, et.al UW, EMNLP 2017)

- LSTM
- Attention

for every subsequence, they come up with some span representation $g_i = [x^*_{start}, x^*_{END}, \hat{x}_i, \phi(i)]$

Attention scores $\alpha_t = \omega_\alpha \cdot FFNN_\alpha(x^*_t)$

$s(i,j) = s_m(i) + s_m(j) + s_a(i,j)$

↑ is that a mention; ↑ Is that a mention; ↑ Do they look coreferent

$s_m(i) = \omega_m \cdot FFNN(g_i)$

$s_a(i,j) = \omega_a \cdot FFNN_a([g_i g_j, g_i \circ g_j, \phi(i,j)])$

- Intractable to score every pair of spans
  - $O(T^2)$ spans of text in a document
  - $O(T^4)$ runtime

Clustering Based Approach

- Coreference is a clustering task. Let's use a clustering algorithm (agglomerative clustering)
- Start with each mention in it's own ~~cluster~~ singleton cluster
- Merge a pair of clusters at each step
  → Use a mode to score which cluster merges are good.
- Mention pair decision is difficult
- Cluster-pair decision is easier

Evaluation: MUC, CEAF, LEA, B-CUBED, BLANC

Conclusion:

- Coreference is a useful, challenging and linguistically interesting task
- Systems are getting better rapidly, largely due to better neural models

Lecture 17: Multi-task Learning

Richard Socher (Chief scientist, Salesforce)

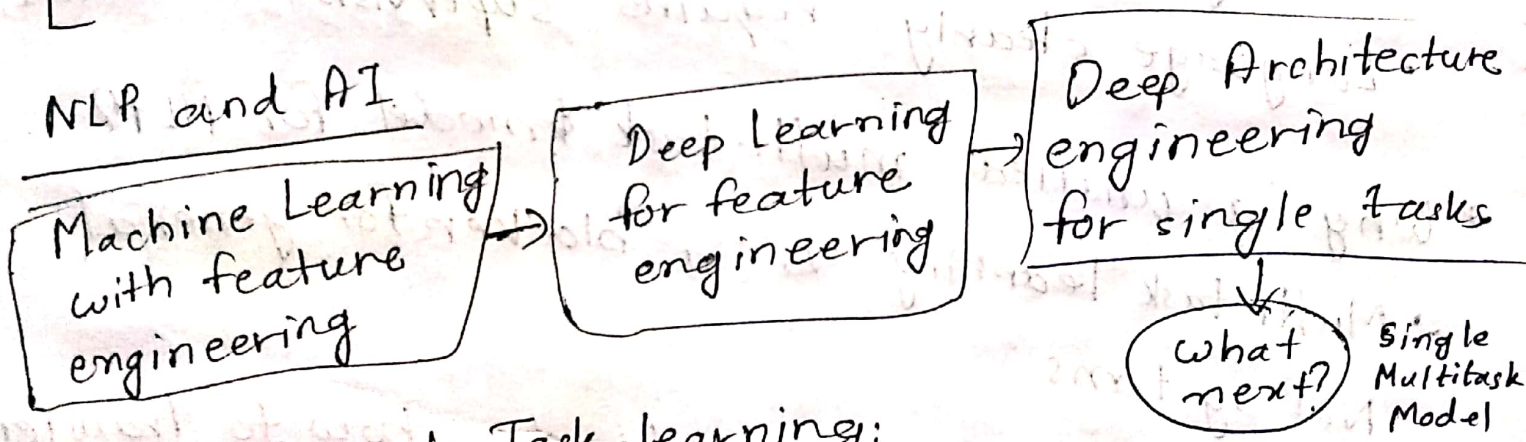
DecaNLP

The Natural Language Decathlon : Multi Learning as Question Answering

Joint work with Bryan McCann, Nitish Keshar and Caiming Xiong

[10 tasks in one model]

NLP and AI



Limits of Single Task learning:

→ Great performance improvements in recent years given {dataset, task, model, metric}

→ We can hill-climb to local optima as long as |dataset| > 1000×C

→ For more general AI, we need continuous learning in a single model instead

→ Models typically start from random or are only partly ↓ pretrained

- Why has weight & model sharing not happened as much in NLP?

→ NLP requires many types of reasoning: logical, linguistic, emotional visual and others

→ requires short and long term memory

→ NLP had been divided into intermediate and seperate tasks to make progress

- A single unsupervised ~~no~~ task cannot solve it all
- Language clearly requires supervision in nature

* Why a unified multi-task & model for NLP?

→ Multi-task learning is a blocker for general NLP systems

→ Unified models can decide how to transfer knowledge (domain adaption, weight sharing, transfer and zero shot learning)

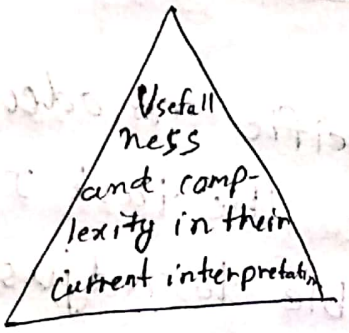
→ Unified, multi-task models can
  - more easily adapt to new tasks
  - make deploying to production X times & simpler
  - lower the bar for more people to solve new tasks
  - Potentially move forward continual learning

How to express many NLP tasks in the same framework?

- Sequence tagging: NER, aspect specific sentiment
- Text classificatin: dialogue state tracking, sentiment classification
- Seq2Seq: machine translation, summarization, question answering

3 equivalent supertasks of NLP:

Language Modelling
Question Answering
Dialogue



decaNLP

(Question Answering)

- pharase extraction from context (Question Answering)
- translation
- summary
- NLI
- sentiment classification
- semantic role labeling
- Relation extraction
- Dialogue state tracking
- SQL translation (Semantic pasing)
- Winograd schema and anaphora resolution (Common sense reasoning)

- Meta Supervised Learning: from $\{x, y\}$ to $\{x, t, y\}$ (t is the task)
- Use a question, q, as a natural description of the task, t, to allow the model to use linguistic information to connect tasks
- y is the answer to q and x is the context necessary to answer q

Designing a model for decaNLP

Specifications

- No task-specific modules or parameters because we assume the task ID is not available
- Must be able to adjust internally to perform disparate tasks.
- Should leave open the possibilities of zero-shot inference for unseen tasks.

★ Start with a context
★ Ask a question
★ Generate the answer one word at a time by
  - Pointing to a context
  - Pointing to question
  - Or choosing a word from an external vocabulary

- Pointer switch chooses ~~betwe~~ among those three options

Multitask Question Answering Network (MQAN).

Initial Encoding → co-attention → Transformer Layer ×2 → Final Encoding

Datasets	Metrics	Task
SQuAD	nf1	QA
IWSLT En-De	BLEU	MT
CNN/Daily Mail	Rouge	sumarization
MultiNLI	EM	Natural Language Inference
SST2	EM	Sentiment
QA-SRT	nf1	Semantic Role Labeling
QA-ZRE	CF1	Relation Extraction
WOT	dsEM	Goal-Oriented Dialogue
WikiSQL	lfEM	semantic Parsing
Winograd Schemas	EM	Pronoun Resolution

decaScore = sum of all metrics

- Transformer layers yield benefit in single-task and multitask learning
- QA and SRL have strong connection
- Pointing to the Question is essential
- Multitasking helps zero-shot

- There is a gap between the combined single-task models and the sig single multitask model.

* Neural architecture search

Reinforcement learning to figure out what layer comes next

Training strategy

Fully joint → Round Robin baild minibatch out of all tasks.

Anti-curriculum pretraining:
train on the harder tasks and add the simpler tasks later on

Closing the Gap

i) Fully joint training
ii) Anti-curriculum pretraining
iii) GloVE to CoVe
iv) including more tasks anti-curriculum pretraining
v) oversampling IWSGLT

When MQAN points

- Answers are correctly copied from either context or question
- No confusion over which task the model should perform or which output space to use.

Zero shot Domain Adaptation on pretrained MQAN

- achieves 80% on Amazon and Yelp reviews
- Achieves 62% on SNLI (87% with fine-tuning, a 2-point gain over random initialization)
- BERT is not the silver-bullet to multi-task learning

Most exciting thing would be combining multiple tasks.

- If our model can translate english to german and summarise, why can't be want a german summary? (Interesting question)

decaNLP is a benchmark for generalized NLP

- Train single question answering model for multiple NLP tasks

Framework for tackling

→ more general language understanding

→ multi-task learning

→ domain adaption

→ Transfer learning

→ weight sharing, pre-training, fine-tuning

→ zero-shot learning