Pretraining a Conditioned Generation Architecture

Or Making BERT, BART, and Other Monsters Talk

Michele Bevilacqua

bevilacqua@di.uniroma1.it
SapienzaNLP
Sapienza University of Rome

Table of Contents

1. Background
   - Neural Networks 101
   - Generation
2. General issues in generation
3. Pre-training and Transfer Learning
   - Introduction
   - How to adapt to your own task
4. Conclusion

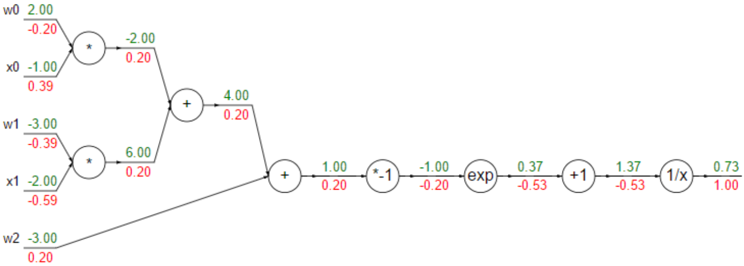
SAPIENZA NLP

Table of Contents

1. Background
   - Neural Networks 101
   - Generation
2. General issues in generation
3. Pre-training and Transfer Learning
   - Introduction
   - How to adapt to your own task
4. Conclusion

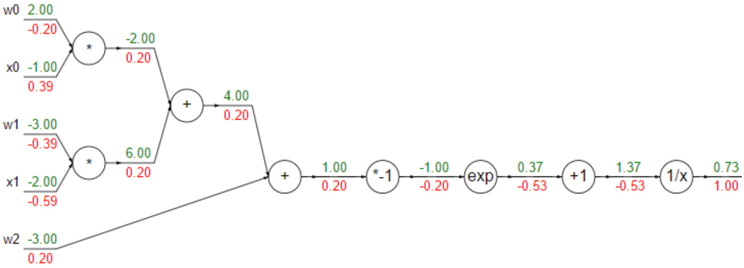
How does a neural network perform a task? There are two important mechanisms:

1. **Forward pass**: application of operations to transform **input numbers** (x_0, x_1) in some other numbers. Operands which are part of the model are called **parameters** (w_0, w_1, w_2).
2. **Backpropagation**:



How does a neural network perform a task? There are two important mechanisms:

1. **Forward pass**:
2. **Backpropagation**: a measure of error (**loss**) is computed between the model output (**prediction**) and the **expected output**. The loss is used to **adjust the weights** in order to bring the prediction closer to the expected values.



Discrete outputs

In Natural Language Processing, we deal with discrete data, which are not naturally translatable into numbers:

- **Inputs:**
  - **Words** (or subdivision thereof) are opaque symbols.
    - Each word is a vector, randomly initialized and refined with backpropagation.
  - **Sentences** or documents are sequences of opaque symbols.
    - We use models that are able to deal with arbitrarily-sized sequences of vectors: Recurrent Neural Networks (RNN) or Transformers.
- **Outputs:**
  - **Word-level**. E.g. PoS tags NOUN, VERB, ADJ.
  - **Sequence-level**. E.g. positive or negative sentiment of a review.

SAPIENZA NLP

Classification *vs.* Generation

Classification

When the task is **discriminatory** we build the architecture such that

- it outputs a vector $\hat{y}$ with as many values as there are output classes;
- the sum of the values y_1 to y_n is 1;
- each value models the probability of a given class given the input;

So, maybe, in the sentence x *The cat* **sat** *on the mat*:

- for the word *sat* $\hat{y}_3 = 0.9$ may correspond to $P(VERB|x)$.
- for the word *sat* $\hat{y}_{10} = 0.01$ may correspond to $P(ADV|x)$.

SAPIENZA NLP

Classification *vs.* Generation

But what if, instead of choosing an option like in PoS tagging, we have to **generate** an entirely new utterance?
Given the sentence x '`The cat sat on the mat`' we train the model to produce '`and was indeed very fat`'.

Classification *vs.* Generation

Generation

1. we produce a next-token distribution given some history h.

Classification *vs.* Generation

Generation

1. we produce a next-token distribution given some history h.
2. we sample from $P(\cdot|h)$ a token $\hat{x}_i$.

SAPIENZA NLP

Classification *vs.* Generation

Generation

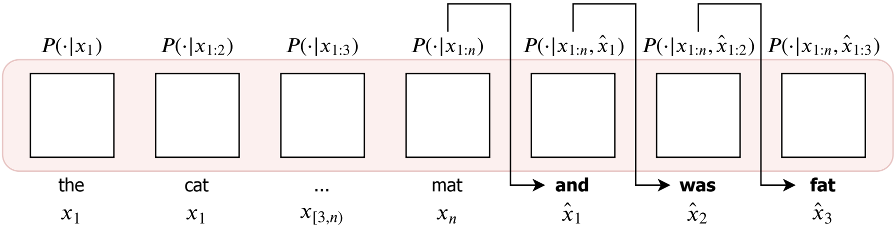
1. we produce a next-token distribution given some history h.
2. we sample from $P(\cdot|h)$ a token $\hat{x}_i$.
3. if $\hat{x}_i$ is an end of sentence special token, we stop, else:

SAPIENZA NLP

Classification *vs.* Generation

Generation

1. we produce a next-token distribution given some history h.
2. we sample from $P(\cdot|h)$ a token $\hat{x}_i$.
3. if $\hat{x}_i$ is an end of sentence special token, we stop, else:
4. we add $\hat{x}_i$ to the history h and go back to step 1.



SAPIENZA NLP

Table of Contents

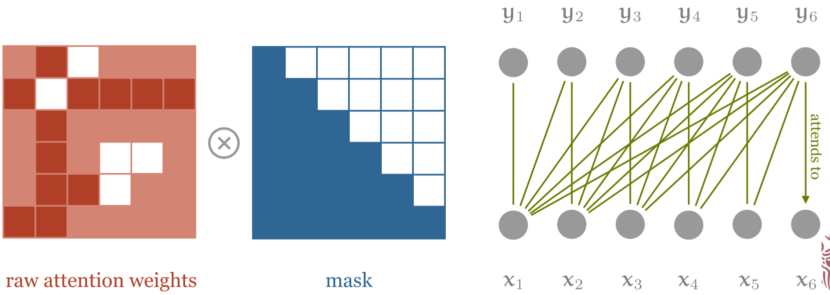
1. Background
   - Neural Networks 101
   - Generation
2. General issues in generation
3. Pre-training and Transfer Learning
   - Introduction
   - How to adapt to your own task
4. Conclusion

SAPIENZA NLP

Decoder Architectures

The generative architecture must be **directional**. That is, it must be able to condition the *i*th prediction only on h_{i-1} and not on future inputs.

- LSTMs (and RNNs in general) naturally satisfy this condition.
- With Transformer layers a $-\infty$ directional mask has to be used to block attention over future tokens.



SAPIENZA NLP

Special tokens

If the incremental generative process needs to include meta-decisions. This can be done by adding special tokens to the vocabulary.

<EOS> Signals to stop generation.

<BOS> Signals to start generation (not really a decision).

Example

`<BOS> the cat sat on the mat <EOS>`

Unconditioned Generation

1. We have seen cases in which the model is asked to **continue** some utterance.
   - $h_0 :=$ `<BOS>` `the cat sat`
   - $h_0 \rightarrow$ `on` $\rightarrow$ `the` $\rightarrow$ `mat` $\rightarrow$ `<EOS>`
2. With this framework, you can also do completely **unconditioned generation**.
   - $h_0 :=$ `<BOS>`
   - $h_0 \rightarrow$ `the` $\rightarrow$ `cat` $\rightarrow$ `sat` $\rightarrow$ `on` $\rightarrow$ `the` $\rightarrow$ `mat` $\rightarrow$ `<EOS>`

SAPIENZA NLP

Conditional Generation

We have seen generation as sequence completion. What if we want to generate given some different different input sequence? e.g.

- Translate EN $\rightarrow$ IT:

  EN: *The cat sat on the mat*

  IT: *Il gatto sedeva sul materasso*

- Abstractive question answering:

  Q: *Where did the cat sit?*

  A: *On the mat*

SAPIENZA NLP

Decoder-only *vs.* Encoder-decoder

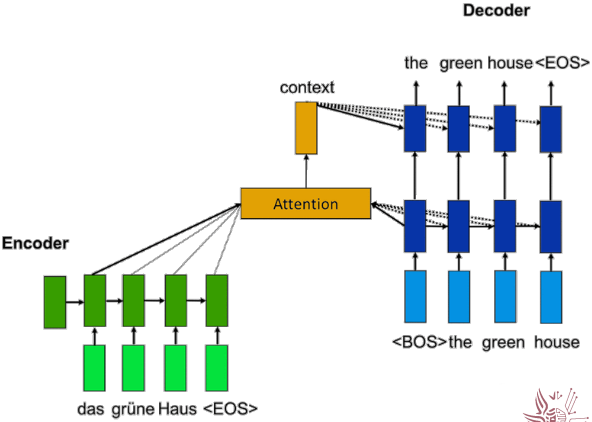
There are two paths you can go by.

1. Decoder-only
   - Concatenate input and output sequences:
     - $h_0 :=$ `<BOS> the cat sat on the mat <EOS>`
     - $h_0 \rightarrow$ `il` $\rightarrow$ `gatto` $\rightarrow$ `sedeva` $\rightarrow$ `sul` $\rightarrow$ `materasso` $\rightarrow$ `<EOS>`
   - Train to complete the output sequence.
2. Encoder-decoder
   - has specialized component for source and targets (with disjoint set of parameters)
   - is trained to generate the target sequence given the source sequence.
     - enc. - $E :=$ `<BOS> the cat sat on the mat <EOS>`
     - dec. - $h_0 :=$ `<BOS>`
     - dec. - $\langle E, h_0 \rangle \rightarrow$ `il` $\rightarrow$ `gatto` $\rightarrow$ `sedeva` $\rightarrow$ `sul` $\rightarrow$ `materasso` $\rightarrow$ `<EOS>`

SAPIENZA NLP

The Encoder-decoder architecture

1. Encoder encodes source (E).
2. Decoder encodes target and predicts (expanding h_0 to h_n).
3. Decoder conditioned on Encoder with attention.



SAPIENZA NLP

Decoder-Only or Encoder-Decoder?

So, which should I use? Decoder-only or Encoder-Decoder?
We are going to try to give an answer to the question later...

Searching the hypothesis space

- The generation process can be seen as modeling a search in an unbounded hypothesis space.

$$P(S|h_0) = \prod_{i=1}^{|S|} P(w_i|h_{i-1})$$

- Computing $P(\cdot|h_0)$ is untractable. Probabilities over an infinite set of possible outputs:
  - `<BOS> the cat sat on the mat <EOS>`
  - `<BOS> the fat cat sat on the mat <EOS>`
  - `<BOS> the fat fat cat sat on the mat <EOS>`
  - `<BOS> the fat fat fat cat sat on the mat <EOS>`
  - `<BOS> the fat fat [...] fat cat sat on the mat <EOS>`

Pruning the search space

- **Max length**: stop searching when output sequence reaches t tokens.
  - Finite search space, but still $|V|^t$ possible sequences!.
- **Min length**.
- **Random decoding** from the output distribution.
  - We pick $w_i \backsim P(\cdot|h_{i-1})$
  - Can use temperature τ parameter to make softmax flatter or more skewed.

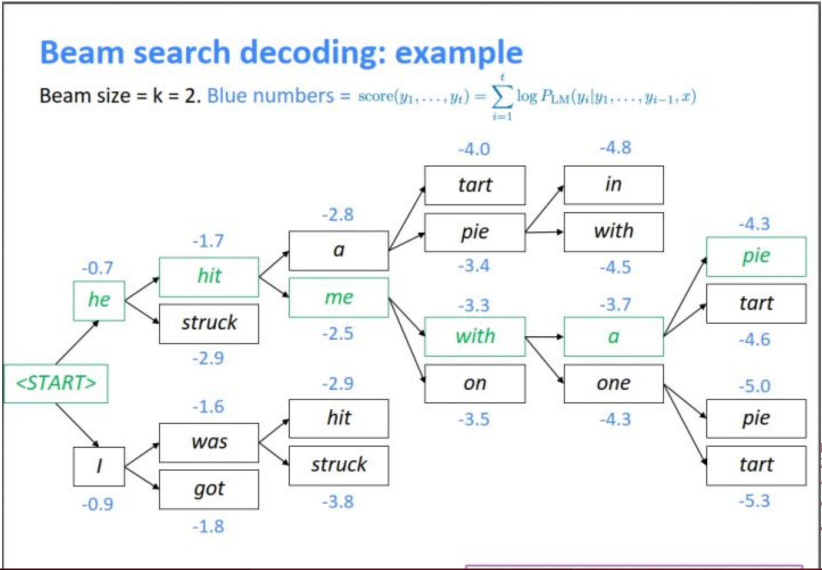
$$P(w|h) = \frac{exp(Z_w/\tau)}{\sum_{w' \in V} exp(Z_{w'}/\tau)}$$

  - Non-deterministic: each run yields different results.
- **Greedy decoding**: choose the locally optimal output.
  - We always pick $\text{argmax}_{w \in V} P(w_i|h_{i-1})$

SAPIENZA NLP

Pruning the search space - Greedy Search

Simply follow the maximum log probability path.



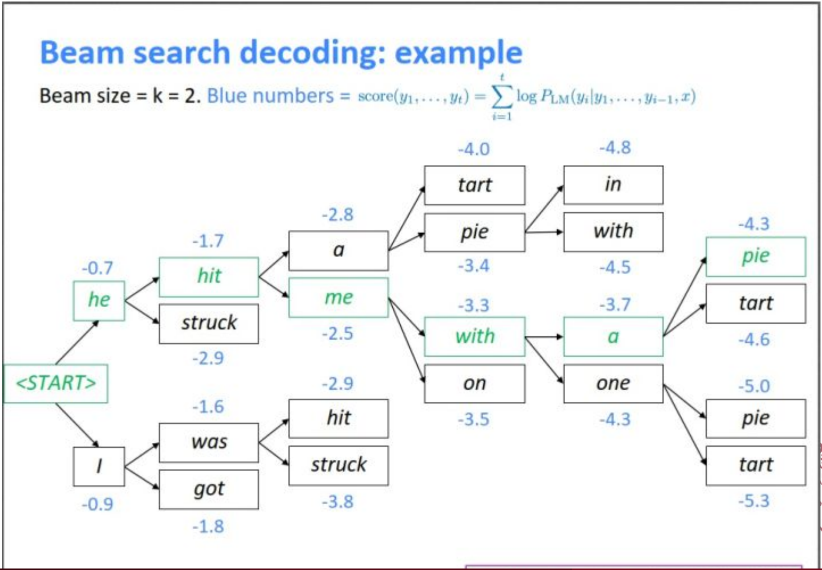
Pruning the search space - Beam Search

Simple Greedy decoding may not yield the optimal solution - the output sequence with highest probability. We cannot explore all possible paths, there are simply too many of them. Solution: we expand the paths which *locally* seem more promising. This is **beam search**.

- the k of best hypothesis are expanded at each search timestep.
- the k best hypothesis that cumulatively reach some probability threshold.
- all the hypothesis whose probabilities exceed a certain threshold.

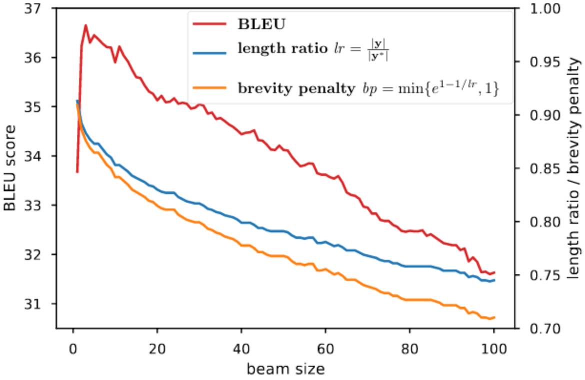
Pruning the search space - Beam Search/II

k-greedy search: at each step, expand the k-best paths.



Known issues with (histogram) beam search I

- Expensive to compute!
- Biased towards shorter sequences. Probability is non-increasing.
- Larger beam produces worse outputs [Yang et al., 2018].



Known issues with (histogram) beam search II

- Neural text degeneration [Holtzman et al., 2019, Welleck et al., 2019]:

prompt	`[...] starboard engines and was going to crash. "We're going in,"`
generation	`he said. "We're going to crash. We're going to crash. We're going to crash. We're going to crash. We're going to crash. We're going to crash. We're going to crash. We're going to`

SAPIENZA NLP

Solutions to the beam search problem I

Search Error vs. Model Error

- Search error is the failure to find the model's optimal score.
  - More beams result in lower search error.
- Model error is the failure to produce the optimal value according to the evaluation metric.

Causes

There is a strong disconnection between training and inference in generation (except for LM):

- The objective function optimized during training (MLE) and the evaluation metric (BLEU, ROUGE, METEOR, Smatch, BERTScore, whatever) are different
- Loss is word-level, evaluation metric is sequence-level;
- **Exposure bias**: the model is never exposed to its own mistakes due to teacher forcing.

SAPIENZA NLP

Solutions to the beam search problem II

Solutions:

- Heuristics: length penalty, coverage penalty;
- Use a better objective, but still $=/=$ from the evaluation metric (Unlikelihood Training) [Welleck et al., 2019]
- Rerank outputs according to the evaluation metric, e.g. with Minimum Bayes Risk Reranking [Liu et al., 2018].
- Optimize the evaluation metric directly [Kreutzer, 2018, Wu et al., 2018, Choshen et al., 2020]
  - Minimum Risk Training
  - REINFORCE; MIXER [Ranzato et al., 2016].

SAPIENZA NLP

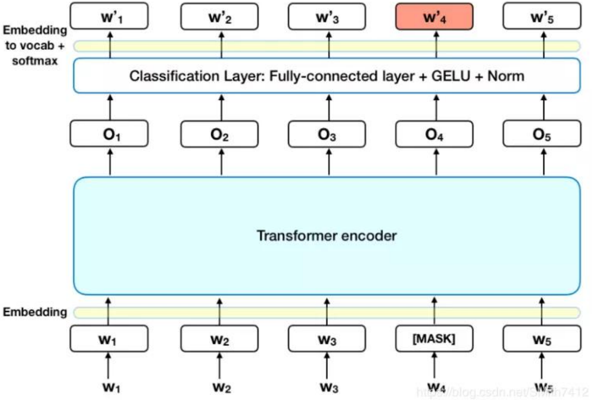
Table of Contents

1. Background
   - Neural Networks 101
   - Generation
2. General issues in generation
3. Pre-training and Transfer Learning
   - Introduction
   - How to adapt to your own task
4. Conclusion

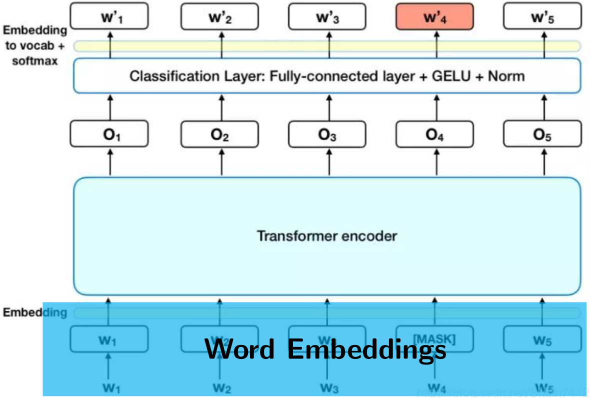
SAPIENZA NLP

Contextualized Embeddings: Architectural Similarities

Most NLP discriminative architectures look pretty much the same...

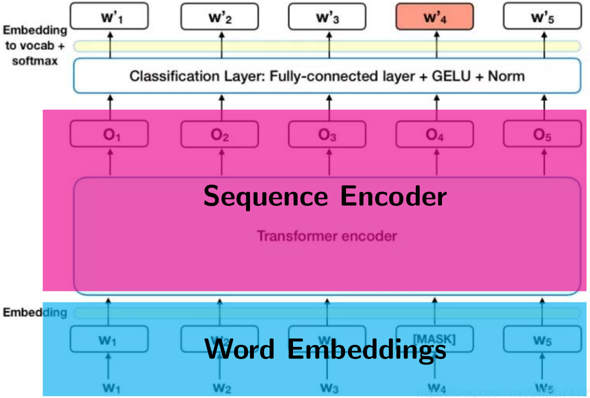


Most NLP discriminative architectures look pretty much the same...



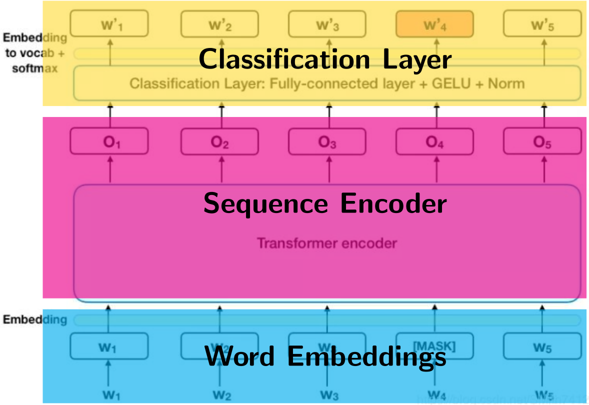
Contextualized Embeddings: Architectural Similarities

Most NLP discriminative architectures look pretty much the same...



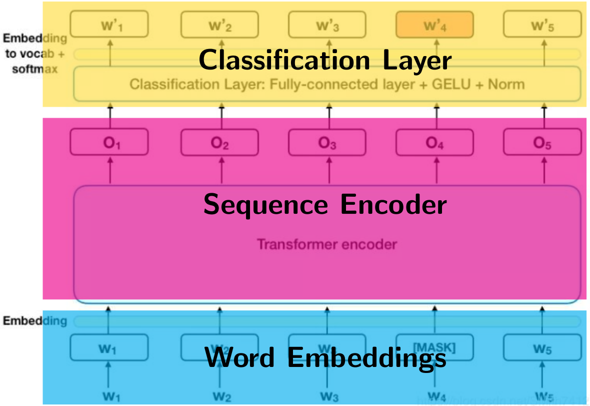
Contextualized Embeddings: Architectural Similarities

Most NLP discriminative architectures look pretty much the same...



Contextualized Embeddings: Architectural Similarities

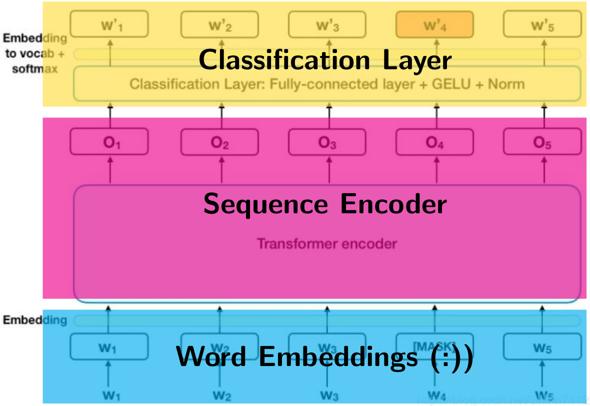
Most NLP discriminative architectures look pretty much the same...



... yet before 2018 only word embeddings were pre-trained.

Contextualized Embeddings: Architectural Similarities

Most NLP discriminative architectures look pretty much the same...



... yet before 2018 only word embeddings were pre-trained.

Transfer Learning through pre-training

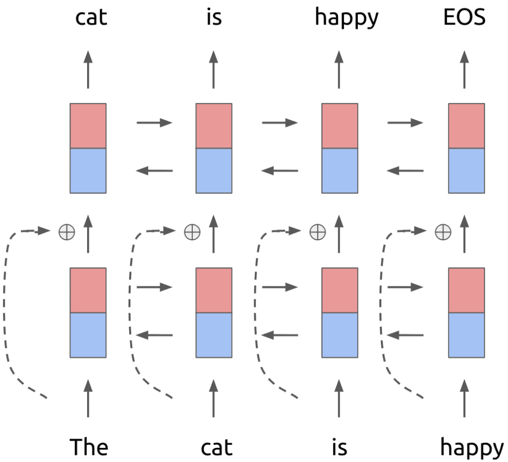
- Since modern neural NLP architecture are modular and share so many components, it makes sense to reuse the knowledge that is acquired on a data-rich task.
- The task that has proven to be most successful for this task is that of **language modeling**.
- Many different flavors of language modeling have been used:
  - Autoregressive Language Modeling (ELMo, GPT2) (also known as vanilla Language Modeling or Causal Language Modeling)
  - Masked Language Modeling (BERT, RoBERTA) (also known as denoising or Cloze task)

Pre-trained models can be exploited in two different ways:

1. **Freeze then embed:** you freeze the weights of the model, use it to produce embeddings which are then fed to a task-specific architecture
2. **Fine-tune:** you remove the prediction layers of the pre-trained model, stack task-specific prediction layers and backpropagate through everything

Causal Language Modeling I

Language modeling has been successfully used as pretraining task. Learning to predict the next word in the sequence requires the model to extract knowledge about morphology, syntax, semantics etc.



Causal Language Modeling II

pro Pre-trained models are naturally auto-regressive.

con Models are directional!

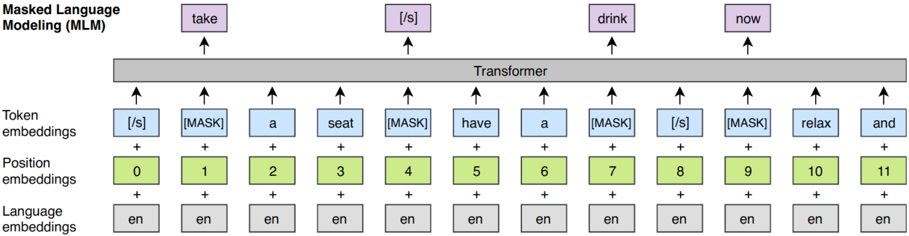
con No explicit Encoder-Decoder architecture.

Models:

- ELMo
- GPT
- GPT-2

Masked Language Modeling I

An alternative to Causal Language Modeling was found in Masked Language Modeling, in which some of the tokens are masked and the model has to reconstruct them.



Masked Language Modeling II

pro Models are bidirectional!

pro Much stronger performances.

con Not auto-regressive

con No explicit encoder-decoder architecture.

Models:

- BERT
- XLM
- RoBERTa

Hybrid CLM/MLM

pro The source sequence is bidirectionally encoded.

pro The target sequence is auto-regressively generated.

con Still no explicit encoder-decoder architecture.

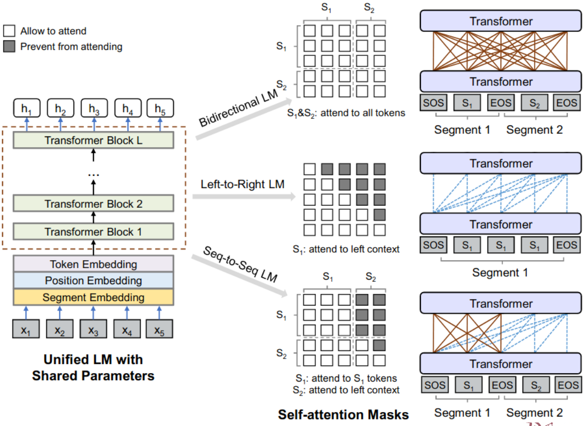
Models:

- XLNet (not going to talk about it)
- UniLM (v1)
- UniLM (v2)

SAPIENZA NLP

Hybrid CLM/MLM - UniLM (v1)

- Decoder-only
- Exploits the flexibility of Transformer's attention masking to train a multitask model:
  - Bidirectional LM (MLM)
  - Directional LM (CLM)
  - Sequence completion (CLM) (allows bidirectional encoding)



SAPIENZA NLP

Hybrid CLM/MLM - UniLM (v2)

Very recently, Microsoft presented a second iteration of UniLM.

- Very complex model - not going to go into details about it.
- Uses a partially randomized factorization order like in XLNet.
- The factorization order uses token spans.
- They use two kinds of masks to train

SAPIENZA NLP

Notes about UniLM:

- Microsoft has released its own framework for fine-tuning UniLM.
- No integration with the `transformers` library is yet available.
- Only a `base` UniLM v2 exists.

SAPIENZA NLP

Encoder-Decoder Architecture

All trained with some form of span generation.

- **pro** The source sequence is bidirectionally encoded.
- **pro** The target sequence is auto-regressively generated.
- **pro** Explicit encoder-decoder architecture.
- **con** Twice the parameter count.

Models:

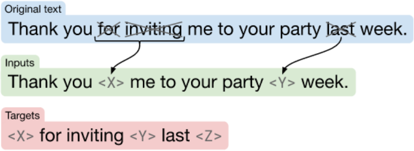
- MASS (not going to talk about it)
- T5
- BART

SAPIENZA NLP

Encoder-Decoder Architectures - T5

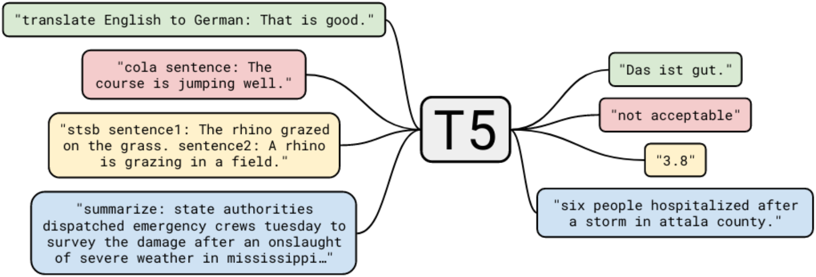
T5 - Text-to-Text Transfer Transformer - [Raffel et al., 2019]
Pre-trained through span prediction.

- Random spans in the input sequence are replaced with placeholders.
- Encoder encodes the masked sequence.
- Decoder predicts the missing bits.
- One of the largest model ever trained - 11 billion parameters. (Smaller models are available though.)



SAPIENZA NLP

T5 has a very interesting approach to fine-tuning. The source encodes both the source string and the downstream task in a template. Fine-tuning is done in multiple tasks at once.



Encoder-Decoder Architectures - T5

Notes about T5:

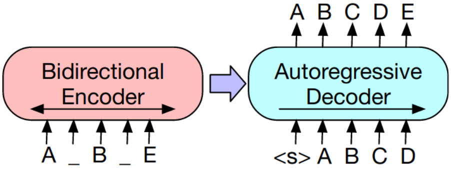
- Training code released (but in `tensorflow`) :(.
- Training code also available in a Colab notebook.
- Checkpoints are compatible with HuggingFace's `transformers`.

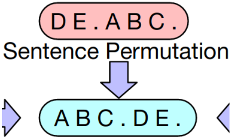
SAPIENZA NLP

Encoder-Decoder Architectures - BART

BART - Bidirectional and Auto-Regressive Transformers - [Lewis et al., 2019]
Pre-trained through token infilling (a kind of span prediction).

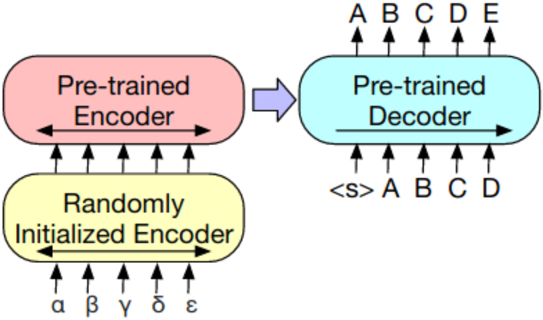
- Random spans in the input sequence are replaced with mask (one per span).
- Encoder encodes the masked sequence.
- Decoder the whole document.
- Combined with sentence shuffling.





BART can be readily fine-tuned for conditional generation tasks. One interesting technique is adopted for NMT.

- They give as input to the pre-trained encoder the output of a randomly-initialized Transformer;
- Foreign language is basically treated as mangled English.



Notes about BART:

- Training code released (in PyTorch, through the `fairseq` library).
- Checkpoints are compatible with HuggingFace's `transformers`.

Performance Comparison

Only a limited comparison is possible between UniLM v1, UniLM v2, T5 and BART. The only dataset on which they all report performances is the CNN/Daily Mail summarization dataset.

		CNN/DM		
		R1	R2	RL
Base	T5-Base	42.05	20.34	39.40
	UniLM v2	**43.16**	**20.42**	**40.14**
Large	T5-Large	42.5	20.68	39.75
	UniLM v1	43.08	20.43	40.34
	BART	**44.16**	**21.28**	**40.9**

- BART is the best option to use!
- If compute is an issue and you are fine with a decoder only architecture, use UniLM v2.

SAPIENZA NLP

Adapting to your conditional generation task!

There are several ways to do it, but not all model families are equally likely to succeed.

- **Trivial techniques:**
  - Produce embeddings (~~CLM~~, MLM, Hybrid, Encoder-Decoder).
  - Fine-tune decoder-only (~~CLM~~, Hybrid)
  - Initialize encoder (~~CLM~~, MLM, Hybrid, Encoder-Decoder)
  - Initialize decoder (~~CLM~~, ~~MLM~~, ~~Hybrid~~, ~~Encoder-Decoder~~)
    - It is usually beneficial only in the low-resource setting!
  - Fine-tune whole Encoder-Decoder (Encoder-Decoder)
- **Non-trivial techniques:**
  - Replacing the input of encoder with another Transformer.
  - Unsupervised seeding (introd. in GPT-2)

SAPIENZA NLP

Trivial Techniques I

We will not spend too much time on the use of embeddings or fine-tuning for conditional generation.

Difference with Transfer Learning in discriminative tasks

- You don't need to have a task specific layer. Subword are very useful with this.
- You may need to modify the vocabulary though, ex. by adding special tokens.
- If you need to substitute completely the input vocabulary of the Encoder, you can check BART's solution for NMT.

SAPIENZA NLP

Trivial Techniques II

The **common wisdom** stays true:

- A lower learning rate limits catastrophic forgetting.
- Use large batches.
- If large batches do not fit in memory, use gradient accumulation.

SAPIENZA NLP

Trivial techniques III

Additionally, there are **some points which are relatively non-controversial**:

- If you can use a pre-trained generative model (Encoder-Decoder or Decoder-only), why use BERT?
- The use of pre-trained models for MT might not be beneficial with big datasets.
- Encoder initialization is better than decoder initialization.

More references

Edunov et al. [2019], Clinchant et al. [2019], Lewis et al. [2019], Raffel et al. [2019], Lample and Conneau [2019], Conneau et al. [2019], Song et al. [2019], Liu et al. [2020]

Decoder-Only *vs.* Encoder-Decoder

Decoder-Only:

- \+ shared params: makes sense when source and target come from similar distributions, e.g. the same language
- \+ shared params: P parameters per layer
- – somewhat lower results with fine-tuning
- – no target spec. params
- – only few pretrained LMs allow bidirectional encoding for source

Encoder-Decoder:

- \+ disjoint params: better suited to handle different languages
- \+ somewhat better results with fine-tuning
- \+ widely used
- \+ more expandable
- – $2P$ params per layer

SAPIENZA NLP

No supervision - Seeding the generation

In the GPT-2 paper, they show that large language models encode general knowledge that can be used to generate answers to any task that can be defined through natural language examples. Basically they:

1. Encode a few source, target pairs as follows:
   - `L'italiano è una lingua bellissima = Italian is a beautiful language`
   - `Non più idee sugli esempi = I am running out of ideas for examples`
2. Concatenate all pairs to history.
3. Concatenate a source on which to condition the generation:
   - `Il gatto sedeva sul materasso =`
4. Press Enter and hope it generates `The cat sat on the mat`

No supervision - Results

Does it work? Kinda:

- 5 BLEU on En-Fr WMT 14.
- Better results on question answering (Natural Questions). Examples:
  - Who wrote the book the origin of species? Charles Darwin
  - Who plays ser davos in game of thrones? Peter Dinklage

No supervision - A silly exploration

N.B. All of the following experiments were performed on `https://transformer.huggingface.co/doc/gpt2-large`. The full GPT-2 is available but does not work.

It-En Translation

seed (1) `L'italiano è una lingua bellissima = Italian is a beautiful language`

seed (2) `Non più idee sugli esempi = I am running out of ideas for examples`

generation `Il gatto sedeva sul materasso = The teacher is running out of time`

SAPIENZA NLP

No supervision - A silly exploration

N.B. All of the following experiments were performed on `https://transformer.huggingface.co/doc/gpt2-large`. The full GPT-2 is available but does not work.

It-En Translation (new attempt)

seed (1) `L'italiano è una lingua bellissima = Italian is a beautiful language`

seed (2) `Non più idee sugli esempi = I am running out of ideas for examples`

generation `Mamma mia = I am my mother`

No supervision - A silly exploration

N.B. All of the following experiments were performed on `https://transformer.huggingface.co/doc/gpt2-large`. The full GPT-2 is available but does not work.

Token reversal

```
  seed (1) they had a cake = cake a had they
  seed (2) he is shivering today = today shivering is he
generation they had a donut = donut a had they
```

N.B. All of the following experiments were performed on `https://transformer.huggingface.co/doc/gpt2-large`. The full GPT-2 is available but does not work.

En-Yoda translation

```
  seed (1) they had a cake = a cake they had
  seed (2) he is shivering today = today shivering he is
generation they had a donut = a donut they had
```

SAPIENZA NLP

N.B. All of the following experiments were performed on `https://transformer.huggingface.co/doc/gpt2-large`. The full GPT-2 is available but does not work.

Definition generation

seed (1) `swag = (slang) cool, hip`

seed (2) `manosphere = blogs, websites and internet resources targeted ad men's right activists`

seed (3) `incel = involuntary celibate`

generation `alt-right = alt-right is a loose collection of white nationalists, white supremacists, and other white supremacists who are opposed to multiculturalism, feminism, and other progressive social movements.`

SAPIENZA NLP

No supervision - A silly exploration

N.B. All of the following experiments were performed on `https://transformer.huggingface.co/doc/gpt2-large`. The full GPT-2 is available but does not work.

Contextual definition generation

seed (1) `the **fan** is not working = a rotating device used to cool off high temperatures`

seed (2) `Tommy is a **man** = a grown-up male person`

generation `i am eating a **dog** = a dog that is eating a human`

SAPIENZA NLP

No supervision - A silly exploration

N.B. All of the following experiments were performed on `https://transformer.huggingface.co/doc/gpt2-large`. The full GPT-2 is available but does not work.

Generative constituency parsing

```
  seed (1) This is a wug = [ S [ NP [ N This ] ] [ VP [ V
           is ] [ NP [ Det a ] [ N wug ] ] ] ]
  seed (2) The boy eats apples = [ S [ NP [ Det the [ N boy
           ] ] [ VP [ V eats ] [ NP [ N apples ] ] ] ]
generation The dog ate the cat = [ S [ NP [ N cat ] ] [ VP
           [ V eats ] [ NP [ N dog ] ] ] ]
```

No supervision - A silly exploration

N.B. All of the following experiments were performed on `https://transformer.huggingface.co/doc/gpt2-large`. The full GPT-2 is available but does not work.

Generative AMR parsing

seed (1) `My drawing was not a picture of a hat . = [...]`

seed (2) `They always need to have things explained . = [...]`

generation
```
So then I chose another profession , and learned
to pilot airplanes . =
(n / pilot-01
:ARG0 (t / they )
:ARG1 (e / explain-01)
:time (a / always ))
```

SAPIENZA NLP

Table of Contents

1. Background
   - Neural Networks 101
   - Generation
2. General issues in generation
3. Pre-training and Transfer Learning
   - Introduction
   - How to adapt to your own task
4. Conclusion

SAPIENZA NLP

Concluding remarks

- There is a widening array of pre-trained models for conditional generation.
- Unsupervised techniques deserve way more attention than what they are receiving.
- The full implications of T5's natural language encoding of tasks have not yet been explored.

Thank you! Questions?

References I

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eCw3EKvH.

Stéphane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. On the use of BERT for neural machine translation. In Alexandra Birch, Andrew M. Finch, Hiroaki Hayashi, Ioannis Konstas, Thang Luong, Graham Neubig, Yusuke Oda, and Katsuhito Sudoh, editors, *Proc. of EMNLP*, pages 108–117, 2019. URL https://doi.org/10.18653/v1/D19-5611.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *CoRR*, abs/1911.02116, 2019.

Sergey Edunov, Alexei Baevski, and Michael Auli. Pre-trained language model representations for language generation. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proc. of NAACL-HLT*, pages 4052–4059, 2019. doi: 10.18653/v1/n19-1409.

SAPIENZA NLP

References II

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019.

Julia Kreutzer. Rl in nmt: the good, the bad and the ugly. `http://www.cl.uni-heidelberg.de/statnlpgroup/blog/rl4nmt/`, 2018.

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *CoRR*, abs/1901.07291, 2019. URL `http://arxiv.org/abs/1901.07291`.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *CoRR*, abs/2001.08210, 2020.

Yuchen Liu, Long Zhou, Yining Wang, Yang Zhao, Jiajun Zhang, and Chengqing Zong. A comparable study on model averaging, ensembling and reranking in NMT. In *Proc. of NLPCC*, pages 299–308, 2018. URL `https://doi.org/10.1007/978-3-319-99501-4_26`.

SAPIENZA NLP

References III

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. 2016. URL http://arxiv.org/abs/1511.06732.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MASS: masked sequence to sequence pre-training for language generation. In *Proc. of ICML*, pages 5926–5936, 2019. URL http://proceedings.mlr.press/v97/song19d.html.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. *CoRR*, abs/1908.04319, 2019.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. A study of reinforcement learning for neural machine translation. In *Proc. of EMNLP*, pages 3612–3621, 2018. URL https://doi.org/10.18653/v1/d18-1397.

Yilin Yang, Liang Huang, and Mingbo Ma. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proc. of EMNLP*, pages 3054–3059, 2018. URL https://doi.org/10.18653/v1/d18-1342.