

Machine Learning for Natural Language Processing

Transfer Learning with Neural Models for NLP

Lecture 6

Benjamin Muller

INRIA Paris - ALMANACH
benjamin.muller@inria.fr

- Language Model
 - N-gram
 - Neural Language Model
- The Transformer Architecture
- The Sequence to Sequence paradigm

- Transfer Learning
- Word2vec + Task-Specific Architecture
- Language Modelling : Focus on BERT
- Challenges and Limits of NLP
 - The 4 challenges of NLP
 - The 4 questions of any NLP project

Transfer Learning in NLP

Transfer Learning

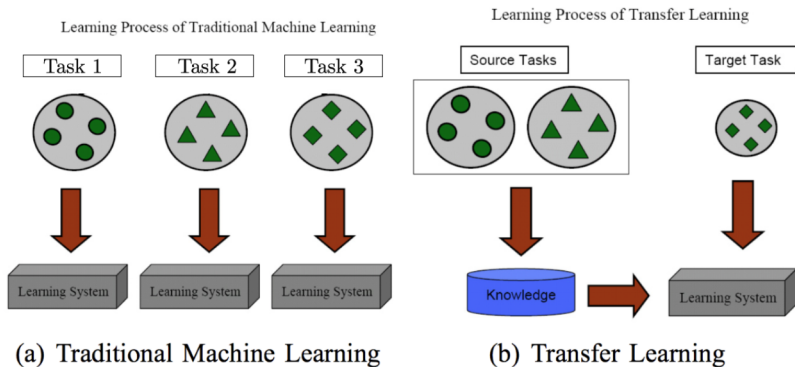


Figure: Pan and Yang (2009)

Machine Learning Framework

- Let A a task, on observation $(X_i, Y_i)_i$, learn a predictive model p_A of $X_i \rightarrow Y_i$

Machine Learning Framework

- Let A a task, on observation $(X_i, Y_i)_i$, learn a predictive model p_A of $X_i \rightarrow Y_i$

Transfer Learning Framework

- Let A and B two tasks. $(X_1, Y_i)_i, (W_i, Z_i)_i$ observations.
 - Learn a model p_A learn a predictive model p_A of $X_i \rightarrow Y_i$
 - Reuse p_A to learn a predictive model p_B of $W_i \rightarrow Z_i$

Machine Learning Framework

- Let A a task, $(X_1, \dots, X_T)_i, (Y_1, \dots, Y_T)_i$ variables,
Goal: Learn a predictive model p_A

$$(X_1, \dots, X_T)_i \xrightarrow{p_A} (Y_1, \dots, Y_T)_i$$

Machine Learning Framework

- Let A a task, $(X_1, ..X_T)_i, (Y_1, .., Y_T)_i$ variables,
Goal: Learn a predictive model p_A

$$(X_1, ..X_T)_i \xrightarrow{p_A} (Y_1, .., Y_T)_i$$

Transfer Learning Framework

- Let A and B two tasks.
 $(X_1, ..X_T)_i, (Y_1, .., Y_T)_i, (W_1, .., W_T)_i, (Z_1, .., Z_T)_i$ variables.
 - Learn a model p_A learn a predictive model p_A of $X \rightarrow Y$
 - **Goal:** Reuse p_A to learn a predictive model p_B

$$p_A, (W_1, ..W_T)_i \xrightarrow{p_B} (Z_1, .., Z_T)_i$$

Different types of Transfer Learning for text

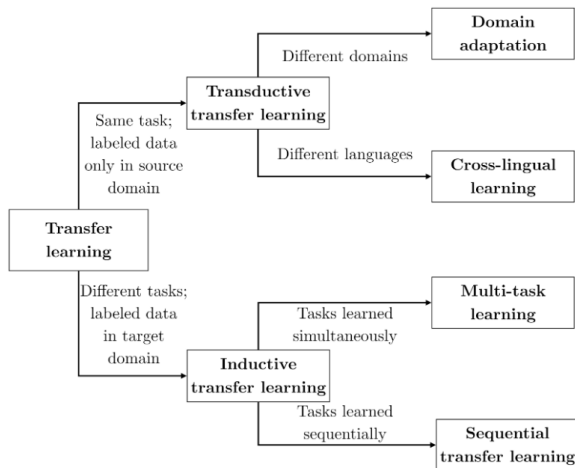


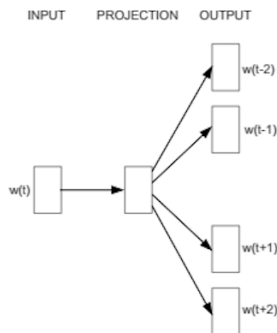
Figure: Ruder (2019)

Sequential Transfer Learning

Sequential Transfer Learning: 2 study cases

- Continuous distributional word vectors used within a task-specific Architecture
- Language Model: Focus on BERT

Task A: Skip-gram model ¹



Skip-gram

By training a model to predict **context** words, we learn an **Embedding** matrix of word vectors

¹cf. Lecture 2

Task B: POS tagging²

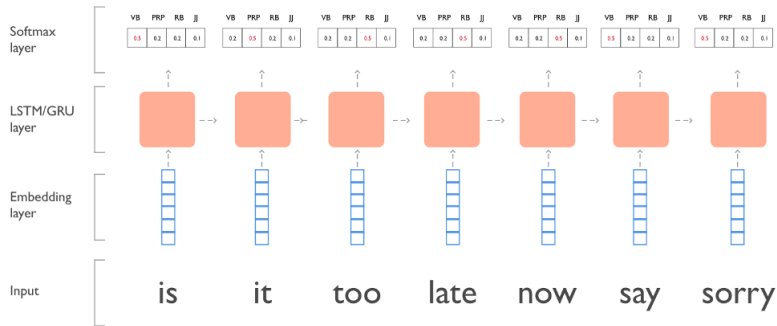


Figure: POS tagging LSTM recurrent network

By default we learn the Embedding layer as any other layers (random initialization)

²cf. Lecture 4

Transferring word vectors to POS tagging

- ① (Pre-)Train a skip-gram model with Embedding Matrix E
- ② Initialize the POS tagger Word Embedding Layer with Embedding Matrix E
- ③ Train the POS tagger with backprop (as seen in Lecture 4)

Transferring word vectors to POS tagging

- ① (Pre-)Train a skip-gram model with Embedding Matrix E
- ② Initialize the POS tagger Word Embedding Layer with Embedding Matrix E
- ③ Train the POS tagger with backprop (as seen in Lecture 4)

Intuition:

- By doing so, the distributional representation learnt during pre-training are adapted, fine-tuned with regard to the downstream POS tagging task.
- The tagger makes use of the word embedding structure to predict POS labels.

- For most NLP tasks, using pre-trained word embedding layer and tuning it gives a significant improvement compared to random initialization

For instance for Name-Entity-Recognition

LSTM NER model	
init	F1
random	80.88
skip-gram	90.33

Table: NER S-LSTM performance CoNLL-03 test set ³

³Lample et al. (2016)

Sequential Transfer Learning:

- Distributional word vectors used within a task-specific Architecture
- Language Models as contextual word vectors: Focus on BERT

BERT

- BERT is a Transformer Architecture
- Trained as a Masked-Language Model
- published and released by Google in October 2019 ()

BERT lead to several breakthroughs in NLP

BERT objective

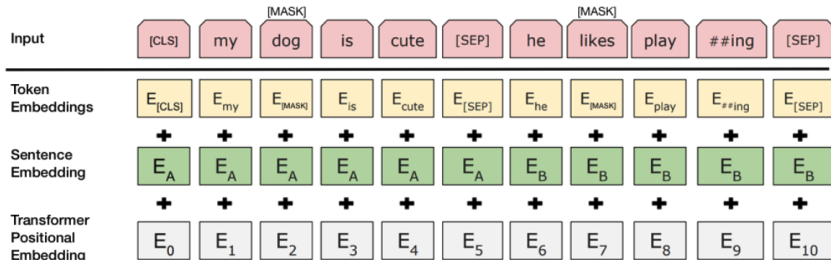


Figure: Bert input and output

- BERT works at the sub-word units
 - No Out-Of-Vocabulary problem
 - Can have on shared vocabulary for multiple languages

- Out-of-Vocabulary problem
- A solution is to work at another level than words
- character level is a solution since there is no UNK at this level
- character level models are extremely slow
- How to build bigger units than character with no UNK?
- These units should be at a **subword level** to cover new words

- Building subword units by segmenting words into subparts
- wishlist for word segmentation algorithm:
- open-vocabulary: encode all words through small vocabulary
- encoding generalizes to unseen words
- small sequence size (e.g., unlike character level)
- Current best solution: segmentation via **byte pair encoding (BPE)**

Byte pair encoding for word segmentation

- BPE is a **bottom-up character merging** algorithm
- compress representation based on information theory

from Sennrich's class

<http://www.inf.ed.ac.uk/teaching/courses/mt/syllabus.html>

Byte pair encoding for word segmentation

- BPE is a **bottom-up character merging** algorithm
- compress representation based on information theory
- BPE algorithm:

from Sennrich's class

<http://www.inf.ed.ac.uk/teaching/courses/mt/syllabus.html>

Byte pair encoding for word segmentation

- BPE is a **bottom-up character merging** algorithm
- compress representation based on information theory
- BPE algorithm:
 0. Split the vocabulary of a dataset at the character level
 - '[hey, hello, et]' \rightarrow 'h','e','y',' ','h','e','l','l','o',' ','e','t'

from Sennrich's class

<http://www.inf.ed.ac.uk/teaching/courses/mt/syllabus.html>

Byte pair encoding for word segmentation

- BPE is a **bottom-up character merging** algorithm
- compress representation based on information theory
- BPE algorithm:
 0. Split the vocabulary of a dataset at the character level
 - '[hey, hello, et]' \rightarrow 'h','e','y',' ','h','e','l','l','o',' ','e','t'
 1. Compute frequencies of consecutive symbol pairs
 - {'h','e'} = 2; {'l','l'} = 1;...

from Sennrich's class

<http://www.inf.ed.ac.uk/teaching/courses/mt/syllabus.html>

Byte pair encoding for word segmentation

- BPE is a **bottom-up character merging** algorithm
- compress representation based on information theory
- BPE algorithm:
 0. Split the vocabulary of a dataset at the character level
 - '[hey, hello, et]' \rightarrow 'h','e','y',' ','h','e','l','l','o',' ','e','t'
 1. Compute frequencies of consecutive symbol pairs
 - {'h','e'} = 2; {'l','l'} = 1;...
 2. replace most frequent consecutive symbol pair ('A','B') with 'AB'
 - merge 'h','e' into 'he'
 - 'he','y',' ','he','l','l','o',' ','e','t'

from Sennrich's class

<http://www.inf.ed.ac.uk/teaching/courses/mt/syllabus.html>

Byte pair encoding for word segmentation

- BPE is a **bottom-up character merging** algorithm
- compress representation based on information theory
- BPE algorithm:
 0. Split the vocabulary of a dataset at the character level
 - '[hey, hello, et]' \rightarrow 'h','e','y',' ','h','e','l','l','o',' ','e','t'
 1. Compute frequencies of consecutive symbol pairs
 - {'h','e'} = 2; {'l','l'} = 1;...
 2. replace most frequent consecutive symbol pair ('A','B') with 'AB'
 - merge 'h','e' into 'he'
 - 'he','y',' ','he','l','l','o',' ','e','t'
 3. repeat step 1-2 until desired dictionary size is reached

from Sennrich's class

<http://www.inf.ed.ac.uk/teaching/courses/mt/syllabus.html>

Byte pair encoding for word segmentation

- BPE is a **bottom-up character merging** algorithm
- compress representation based on information theory
- BPE algorithm:
 0. Split the vocabulary of a dataset at the character level
 - '[hey, hello, et]' \rightarrow 'h','e','y',' ','h','e','l','l','o',' ','e','t'
 1. Compute frequencies of consecutive symbol pairs
 - {'h','e'} = 2; {'l','l'} = 1;...
 2. replace most frequent consecutive symbol pair ('A','B') with 'AB'
 - merge 'h','e' into 'he'
 - 'he','y',' ','he','l','l','o',' ','e','t'
 3. repeat step 1-2 until desired dictionary size is reached
- Works as a preprocessing of the train set

from Sennrich's class

<http://www.inf.ed.ac.uk/teaching/courses/mt/syllabus.html>

- Once the BPE codes are computed, we have a list of merges

$$M = [\text{'e s'} \mapsto \text{'es'}, \text{'es t'} \mapsto \text{'est'}, \dots]$$

- Steps to preprocess a dataset with BPE codes:
 1. split dataset in sequence of characters
 2. apply the sequence of merges to the dataset to form BPE codes
- It is a deterministic process with a unique segmentation

from Sennrich's class

<http://www.inf.ed.ac.uk/teaching/courses/mt/syllabus.html>

BPE: pros and cons

- Pros:
 - Very simple
 - Models trained on BPE segmentations share information between words (e.g., prefix, suffix...)
 - BPE vocabulary is a **best trade-off between vocabulary size and average sequence length**
- Cons:
 - Characters are not necessarily in the final vocabulary → no guarantees open vocabulary (still covers 99.9%)
 - Heuristic, does not work well for all train sets

Other sub-word tokenization algorithm

- WordPiece tokenization
- SentencePiece tokenization

- BERT has been trained on around 60GB of textual data (English Wikipedia, the BookCorpus)

- BERT is a sequence labelling model that use both left and right context for Mask-Language Modelling
- Why not using the full model for a specific task ?
- Constraints:
 - The downstream tasks should be formulated as a sequence labelling/classification task.

BERT architecture

BERT-base

- 110M parameters
- 12 self-attention layers (778 dim)
- 12 heads per layers
- 512 tokens sequence

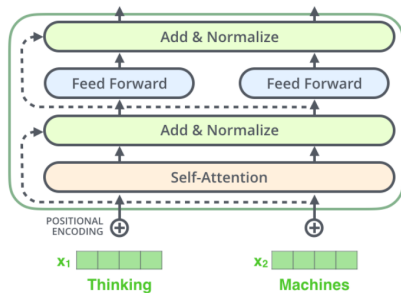


Figure:

Fine-tuning BERT for NER

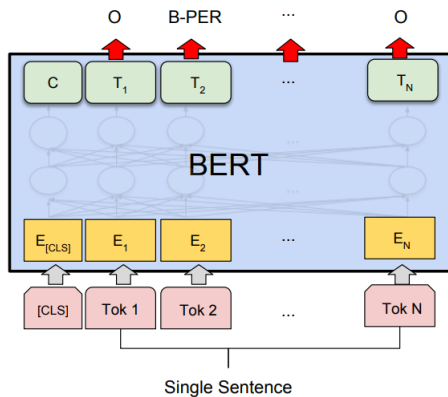


Figure: BERT for NER

Fine-tuning process

- ① Take BERT model
- ② Add an extra task-specific layer (e.g. Dense layer)
- ③ Fine-Tune everything with backpropagation on the new input-output data
- ④ Optimization done with a much smaller learning rate to avoid *catastrophic forgetting*

Hugging Face Transformer library (cf. lab 5)

Transformers provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet...) for Natural Language Understanding (NLU) and Natural Language Generation (NLG) with over 32+ pretrained models in 100+ languages and deep interoperability between TensorFlow 2.0 and PyTorch

NB: The Transformer library is **the** NLP library to work with SOTA models

GLUE: a benchmark for sentence representations

GLUE (Wang et al., 2018) contains 11 tasks covering:

- Single-Sentence Tasks (e.g., text classification)
- Similarity and Paraphrase Tasks
- Inference tasks, i.e., predicting relations between sentences (e.g., coreference, NLI,...)

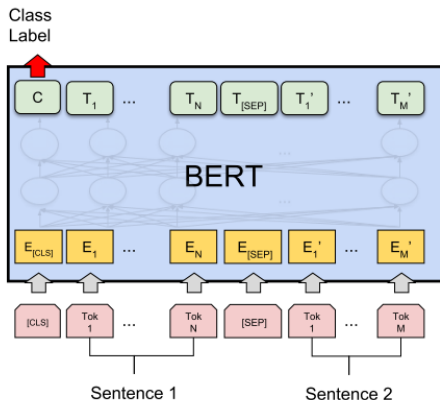
Caveat of GLUE finetuning of models on each task is allowed.

Leaderboard available at <https://gluebenchmark.com/>

Natural language inference (NLI)

- **Goal** predict relation between a premise (P) and an hypothesis (H)
- 3 types of relations:
 - **neutral** the two sentences have no relation
P: A smiling costumed woman is holding an umbrella.
H: A happy woman in a fairy costume holds an umbrella.
 - **contradict** the two statements contradict each other
P: A man inspects the uniform of a figure in some East Asian country.
H: The man is sleeping.
 - **entailment** one of the statement can be inferred from the other
P: A soccer game with multiple males playing.
H: Some men are playing a sport.
- SNLI: <https://nlp.stanford.edu/projects/snli/>

Natural language inference (NLI) with BERT



(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG

Figure: Devlin et al. (2018)

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Table 1: GLUE Test results, scored by the evaluation server (<https://gluebenchmark.com/leaderboard>). The number below each task denotes the number of training examples. The “Average” column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.⁸ BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Figure: Devlin et al. (2018)

- Multilingual BERT-like models (multilingual BERT trained on 104 languages, XLM-R)
- Now monolingual models for many languages (Finish, Dutch, German, Italian, Russian, Chinese,...)
- For French: CamemBERT model ⁴

⁴<https://camembert-model.fr/>

NLP current limits and challenges

Modern NLP models are very efficient to perform on specific tasks on a given distribution (language, domain) Several core challenges remain

- Generalization challenge
Still very hard to generalize across distribution and across tasks
- The *Data Efficiency* challenge (e.g. low resource, cost...)
- The Cost challenge
large model expensive to train and to predict with
- The Interpretability challenge
Best models are black boxes, some use cases require interpretability

The 4 question of any NLP project

The 4 questions of any NLP projects

- ① The Data question
- ② The Modelling question
- ③ The Performance question (speed vs accuracy tradeoff)
- ④ The Ethical question (privacy, biases..)

- ① The Why and What of Natural Language Processing
- ② Representing text with vectors
- ③ Task specific Modeling of Text
- ④ Neural Natural Language Processing
- ⑤ Language Modelling
- ⑥ Transfer Learning

- Project: Design, implement and describe an entire NLP project
- Outcome : Self-contained **notebook** uploaded to **github** and **google colab** + 2 pages latex report

- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.
- Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.
- Ruder, S. (2019). *Neural Transfer Learning for Natural Language Processing*. PhD thesis, National University of Ireland, Galway.
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.