Task agnostic models

Early approaches to NLP via Deep Learning created task-specific architectures.

Some typical tasks

- Classification (sentiment)
- Entailment
- Similarity
- Question answering

These tasks were viewed as independent and trained with task-specific architectures and training sets.

But does there exist

- a single model, trained on an extremely large dataset
- that can be fine-tuned to solve all these tasks ?

There are several impediments that would need to be resolved.

The first is the need for a Universal "API"

- the shape of the input and its format differs between tasks

The second is: a model is trained for a finite (often single) number of tasks.

- How can its parameters encode "knowledge" of unseen tasks ?

Is there a "Universal Task" on which we can train a model

- such that the model can adapt to solving unseen tasks

A Universal Task: Language Modeling

One candidate for the Universal Task is Language Modeling (LM)

- predict the next word

We have already seen how the Language Model was used to create Word Embeddings.

- created representations that encoded relationships between tokens

The Language Model (predict the next word) is a particularly advantageous choice for the Universal Task.

This is because large quantities of training data are readily available

- The data does not need labels
- The label/target for an example is just the next word in the text
    - *semi-supervised* learning, rather than *supervised learning*

There are abundant sources of text sequences

- news, books, blogs, Wikipedia
- not all of the same quality

Models for Language Modeling have also recently been successful in story generation

- author "seeds" the story with a couple of words
- LM extends the story, one word at a time

Predicting the next word in a story requires lots of context ("knowledge").

Consider the task of generating pseudo Shakespeare. The generator needs to "know"

- name of characters
- location specific detail
- plural/singular
- gendered form

Contrast this to Word Embeddings, which are similar

- Embeddings transfer *word-level* concepts
- Language Models transfer *semantic* concepts as well as domain knowledge

This suggest that the Language Model creates representations that encode lots of context/knowledge.

So to use the Universal Model concept to solves a specific task

- Use the Universal API to transform the task input to a common format
- Run the transformed input through a model for Language Modeling
- Use the final latent state of the Language Model as an alternative representation of the original input
    - this is a context/knowledge dense representation of the text
- Run the alternative representation through a shallow, task-specific NN
    - "re-shape" the output for the requirements of the specific task
        - e.g., a Classification task over set C of classes has $||C||$ logits as output layer
    - further transform the representation to adapt to domain

This is basically the setup of Transfer Learning

- Use a pre-trained model
    - obtained at great expense/effort by *someone else*
- Append a task specific head

This approach is called *Supervised Pre-training + Fine-Tuning*

- *Supervised Pre-Training*: the Language Model (e.g., predict the next word)
- *Fine-Tuning*: add a task specific head and fine-tune

A Universal API: Text to text

Because each task has a very specific API (input and output format)

- You have to translate the task-specific format into the format of the Universal Task
- *Text to text* as a universal API
    - transform your task into a "predict the next" task
        - create a "prompt" (context) that describes and encodes your task
        - the Language Model completion of the prompt is the "solution" to your task

For example:

- Consider a Pre-Trained model that performs text completion (predict the next)
- Turn your task into a text completion problem
- See Appendix G (https://arxiv.org/pdf/2005.14165.pdf#page=51) (pages 50+) for examples
    - Note: some of the examples are really "prompts"
        - e.g., seem to be multiple question/answer pairs in addition to a final question with no answer
        - a "prompt" is used for zero/few shot learning
            - the initial question/answer pairs describe the task by multiple example
                - before asking a specific question

Task: Unscramble the letters

Context:	Please unscramble the letters in the word and write that word
	skicts =
Target completion:	sticks

- The "Unscramble the letters task" encoded as "predict the next" word following the "=" sign

Task: English to French

Context:	English: Please unscramble the letters in the word and write that word
	French:
Target completion:	Veuillez déchiffrer les lettres du mot et écrire ce mot

- Translation task encoded as "predict the next" words following the "French:" prompt.

Sometimes the task encodings are not completely obvious (see GPT Section 3.3 (https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf))

- Task: Are two sentences similar ?
    - Issue
        - There is no natural ordering of the two sentences
        - So concatenating the two (with a delimiter) is misleading
    - Solution
        - Obtain two representations of the sentence pair, once for each ordering
        - Add them together element-wise
        - Feed sum into Classifier

- Task: multiple choice questions answering: given context, question plus list of possible answers
    - Solution:
        - Obtain representation for each answer
            - Concatenate (with delimiter): context, questions, answer
        - Feed each representation into a $\mathrm{softmax}$ to obtain probability distribution over answers

GPT: Task encoding

Picture from: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Beyond Transfer Learning: a true Universal Model

The Universal API obviates the need to use a task-specific head to "reshape" the output

- For example: Classification can output labels as text
    - no need for output layer with $||C||$ logits

Do we even need the shallow, task-specific NN appended to the Language Model ?

We will address this topic in the module on Zero Shot Learning.

The future of NLP

From a very practical standpoint

- In the near future (maybe even now) you will not create a new model
- You will use an existing Language Model
    - Trained with lots of data
    - At great cost
- And fine-tune to your task

```
In [1]: print("Done")
```

```
Done
```