So many words !

The size of the vocabulary $\mathbf{V}$ can easily be tens of thousands.

Not only are there many words, there are *variations* of each word

- verbs: past, present and future tense
- nouns:
    - singular, plural
    - gender agreement with subject

"Classical" NLP has developed techniques to convert words to "canonical" form

- stemming: convert word to its root form (root may not be in dictionary)
  - drive, driver, driving $\mapsto$ driv
- lemmatization: convert word to a root that is in dictionary
  - him: $\mapsto$ he
  - took $\mapsto$ take
  - earlier $\mapsto$ early

So one way to reduce the vocabulary size is by using these techniques.

There is still a place for this techniques in Deep Learning.

- [spaCy (https://spacy.io/)](https://spacy.io/) is a very popular toolkit for dealing with text.

It is also possible that Embeddings will create a "dimension" that captures variations common to many words.

- the "plural" dimension

But there is still an issue

- OOV: Out of Vocabulary

No matter how big we make $\mathbf{V}$, there will still be words in use that are not found there.

An elegant solution is *sub-word tokenization*

- break an OOV token into pieces, each of which is in-vocabulary
    - here's $\mapsto$ here, \', s
    - tokenizer $\mapsto$ token, ##izer

Some of the objectives are

- to **not** split frequently occurring words into pieces
    - So that the model consuming the tokens has access to meaningful single word pieces
- to recognize pieces that are common prefix and suffix forms
    - So that the model consuming the tokens can derive the commonality indicated by the suffix
        - e.g. "izer"

One sub-word tokenization method is *Byte Pair Encoding*.

Byte Pair Encoding is derived from techniques for data compression

- Replace long strings that frequently occur in a document with a short symbol
- Use a dictionary to map symbols to their associated strings

Universal model: adapting task-specific inputs

Picture from: http://twimgs.com/ddj/cuj/images/cuj9402gage/fig1.gif"

- Pass 1: AB $\mapsto$ H
- Pass 2: HC $\mapsto$ G

BPE has become fairly common in modern NLP systems

- Balances brevity of character encoding
- With expressiveness of word encoding

In [1]:

```
print("Done")
```

Done