Machine Translation:

Intro:

One of the oldest problems in NLP – large computational techniques

As input a sentence in one languge and the out out in the translation

Input = source, output =target

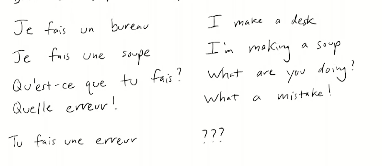
EX french -> English but pretty general techniques

What typ of data:

Paraellel courpus of senetence and their translations

Widely avaible because ethis is what translator s produce as a career

What edge case are we going to se?



We do correspondence - find where “mistake” is and compare

Need to think beyond the word level

Because not a perfect correspondence

Need to think about the phrase level

Big effect : amibugit or many to many translation – need to pick verb tense , might be several correct need to pick

Need to capture phrase level correspondence not word level

2 techniques

Classic machine translation – before neural net – large correspondence dictionary

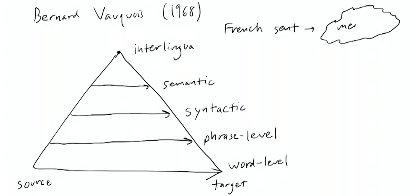
MT: Framework and Evalution

Voquois Triangle

Have a source versus target with different level of translation

Further up the triangell is a higher level of linguistic abstraction

we ant to have an understanding of semantic

inter longua is sent -> meaning -> English

Ie you need to thinak about the meaning and what

The upper layers are hard to deal with

History of MT: word level was first wave, but phrases are the dominate paradigm 2000-2015

Syntax is tough – ppl worked on it , but it wasn’t as well done

Phrases are best right now

Big chunks make translation easier

Two main hurrdels: getting the chunks and translating

Metric called blue: it is the geometric mean of 1-gram 2—gram 3- gram 4-gram precision of output

It also include a brevity penalty

In abstract have a reference translation – how many 4-gram in the refernce

Penalizes short translation

Blue score tends to be low (30-40) human level

This correlates pretty well with human judgements of translation quality – instead know that blue of 33 is better than 30

Can compare across different set of data or different languages

Still a common metric

space\_list = [idx for idx in range(len(text[:-seq\_len-1])) if text[idx] == ' '] ## finding all spaces in the text as the start symbol

chunked\_seqs = [text[t:t+seq\_len] for t in space\_list] ## chunking from the start sysmbol (aka ' ') to the seq length