Character *n*-grams

- Instead of treating each word as unique, why not treat sub-words as unique?
  - Words share characters, and more generally, n-grams of characters!

- Words get factored into (potentially overlapping) character n-grams.
  - Example: *artificial* = <ar, art, rti, tif, ifi, fic, ici, ial, al>

- Allows us to share features between words, as character n-grams are shared across a large number of words.
  - Example: *ben*, *ten*, *happen*, *entropy*, *envisage*, *envy*, *envoy*, *been*, *seen*, *lend*

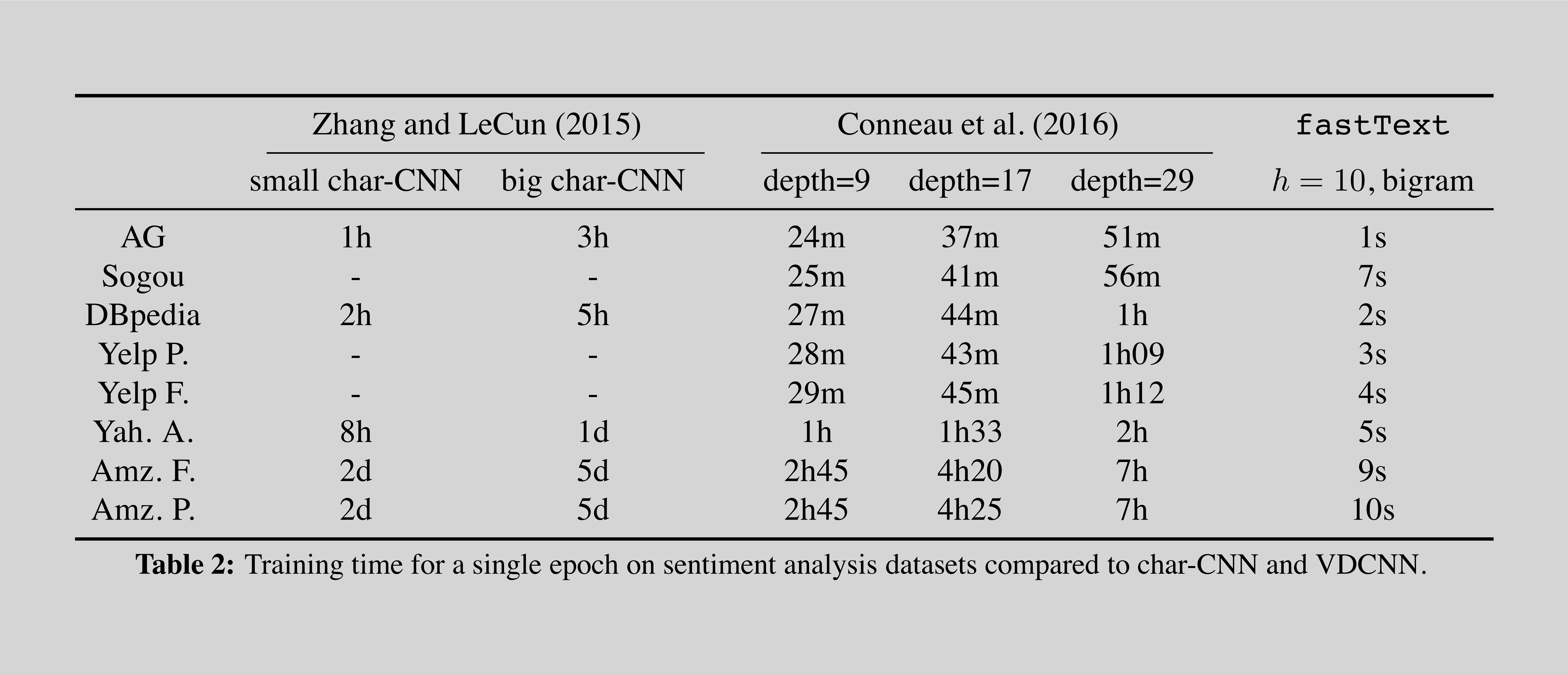
- Inspired by Word2Vec with similar training procedures

- Computational efficient relative to deep learning based methods, while offering very competitive performance on classification and retrieval based tasks.

FastText

- Highly competitive, yet computational efficient, c++ library for both learning word representations and performing text classification.
- Uses multiple *n*-gram features

Joulin et al., 2016, Bag of Tricks for Efficient Text Classification



	Zhang and LeCun (2015)		Conneau et al. (2016)			`fastText`
	small char-CNN	big char-CNN	depth=9	depth=17	depth=29	$h = 10$, bigram
AG	1h	3h	24m	37m	51m	1s
Sogou	-	-	25m	41m	56m	7s
DBpedia	2h	5h	27m	44m	1h	2s
Yelp P.	-	-	28m	43m	1h09	3s
Yelp F.	-	-	29m	45m	1h12	4s
Yah. A.	8h	1d	1h	1h33	2h	5s
Amz. F.	2d	5d	2h45	4h20	7h	9s
Amz. P.	2d	5d	2h45	4h25	7h	10s

Table 2: Training time for a single epoch on sentiment analysis datasets compared to char-CNN and VDCNN.

https://fasttext.cc/docs/en/python-module.html