EMBEDDINGS 2

WHAT IS FASTTEXT?

FastText treats each word as composition of n-grams:

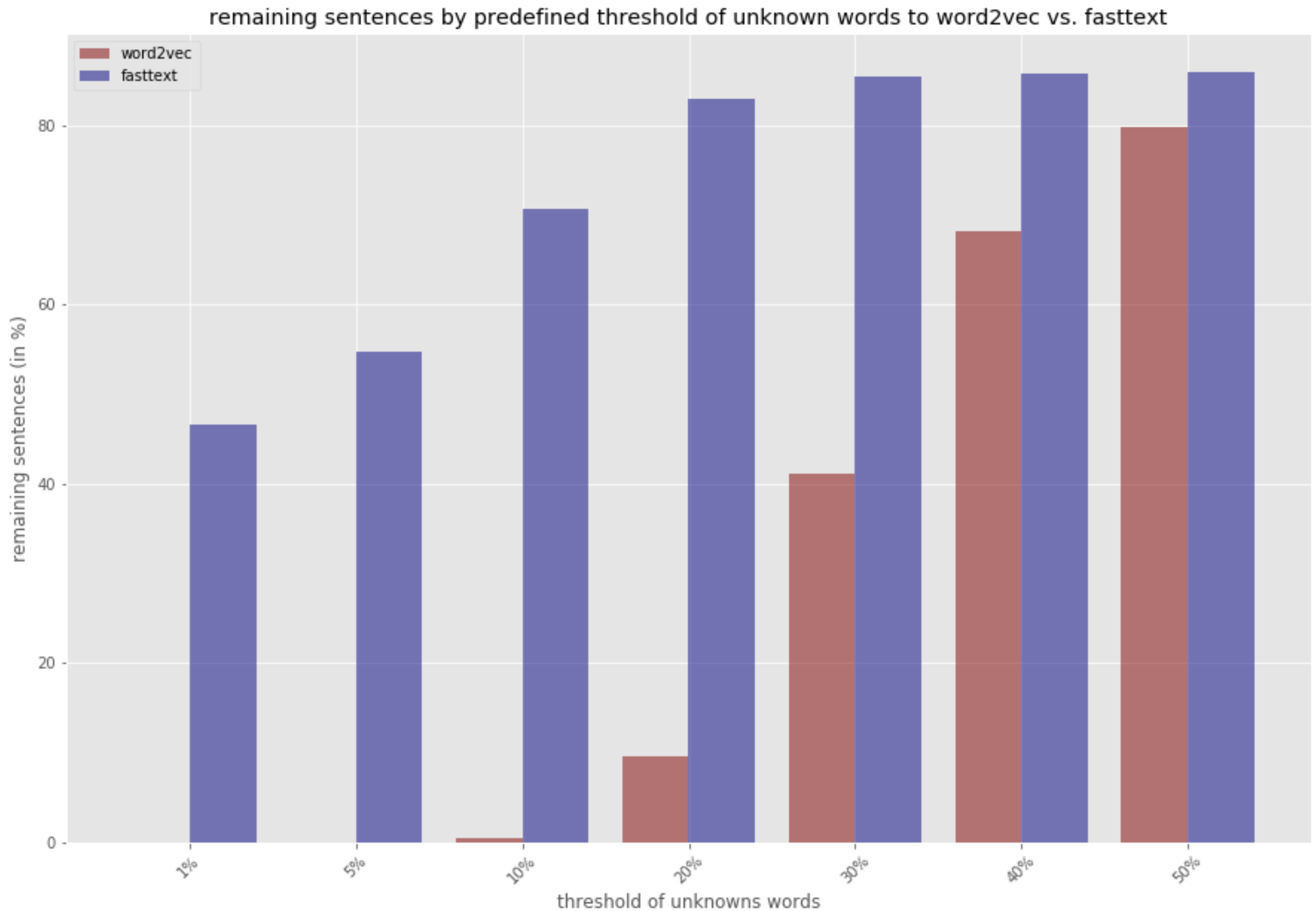
example: ball

"<ba"
"bal"
"ball"
"all"
"ll>"

with smallest n-grams set to 3 and highest to 4

Note that the sequence <all>, corresponding to the word *all*
is different from the tri-gram all from the word *ball*.

WORD2VEC VS. FASTTEXT



FASTTEXT.ZIP

COMPRESSING TEXT CLASSIFICATION MODELS

What?

- reduce memory of text classification model

How?

- quantization
- retraining
- vocabulary pruning
- hashing

QUANTIZATION

The purpose of vector quantization is to compress vectorial data.

general idea:
- find a good set of reference vectors
- replace each data vector by the **index** of its best reference vector

RETRAINING

- retrain the layers occuring after the quantization,
  so that the network can re-adjust itself to the quantization

MEMORY SAVINGS

- compression factor of <u>**10**</u>

  (without any noticeable loss of performance)

- **without RETRAINING:**
  drop in accuracy of 0.5%

VOCABULARY PRUNING

- feature selection problem:
  selecting *K words & ngrams* that preserve
  the highest magnitude to the model

- BUT: some documents won't have any of the K best features

greedy approach

each document -> already covered by retained feature?
if not: add highest magnitude feature to set of retained features

HASHING

- *John likes to watch movies.*
- *Mary likes movies too.*
- *John also likes football.*

Term	Index
John	1
likes	2
to	3
watch	4
movies	5
Mary	6
too	7
also	8
football	9

$$\begin{pmatrix} \text{John} & \text{likes} & \text{to} & \text{watch} & \text{movies} & \text{Mary} & \text{too} & \text{also} & \text{football} \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

MEMORY SAVINGS

- compression factor of up to <u>100</u>

  (dependent on the dataset used)

- complementary with quantization!

combination of both strategies:
model size reduction of up to x1000

ELMO

What?

- word reps that model:
  (1) complex characteristics of word use
  e.g. syntax, semantics
  (2) how this word use vary across context

How?

- word rep as function of entire input sentence
- computed on biLMs with character convolution
- semi-supervised learning

ELMo: Embeddings from Language Models

USE OF ENTIRE INPUT SENTENCE

- each token is assigned a representation

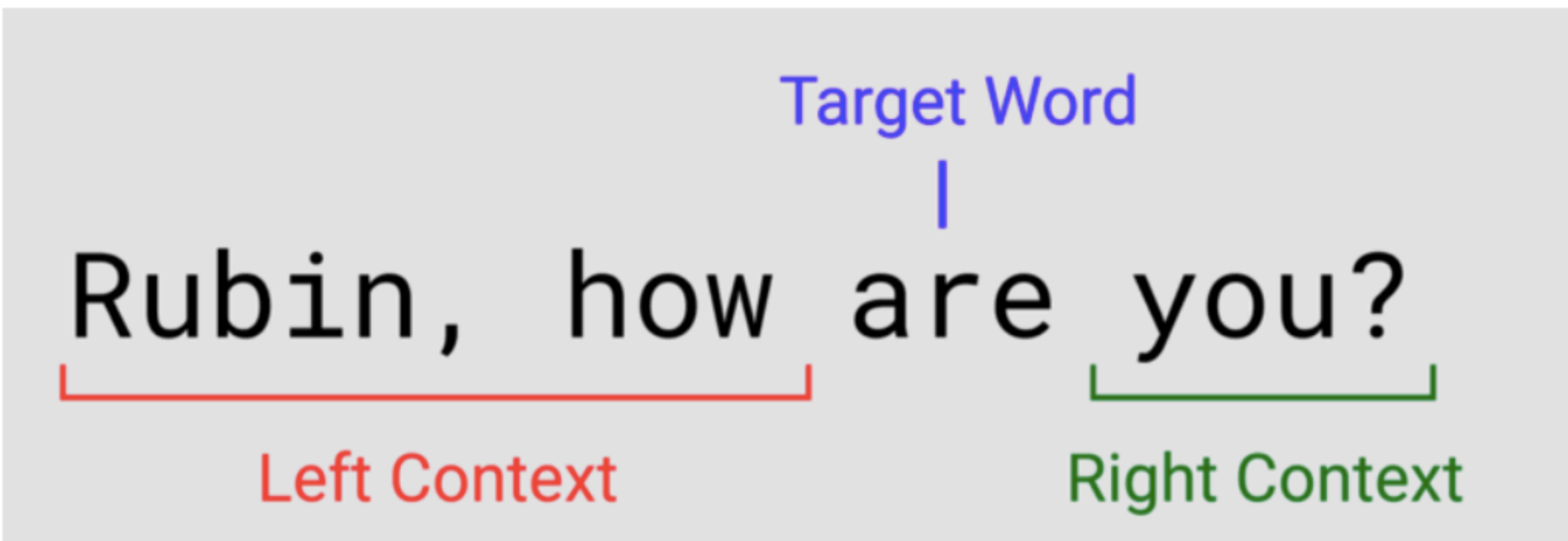
  that is a function

  of the entire input sentence

sentence level embeddings surpass the performance of using word level embeddings

BILM

- vectors are derived from a bidirectional LSTM that is trained with a coupled language Model (LM)



Predicting the word "are" from both left and right contexts.

ELMO REPRESENTATIONS ARE ?DEEP?

- they are a function of all internal layers of a biLM
- learning a linear combination of the vectors stacked behind each input word (improves over just using the top LSTM layer)
- higher-level LSTM states capture context-dependent aspects of word meaning (semantics)
- lower-level LSTM states model aspects of syntax
- exposing these signals allows semi-supervision

FUNCTIONALITY OF ELMO

- ELMo is a combination of the intermediate layer representations in the biLM
- For each token **t**, a biLM with **L** layers computes a set of **2L+1** representations
- ELMo collapses all layers of the biLM into a single vector

$$\mathbf{ELMo}_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

each LSTM layer outputs a context-dependent representation:

$$\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$$

where $j = 1, \ldots, L$

BENEFITS OF ADDING ELMO

- **20%** relative error reduction

- increase the sample efficiency:
  - less parameter updates
  - smaller training set size

  e.g. a Semantic role labeling task
  without ELMo:
  - model reach maximum F1 at **486** epochs of training

  with ELMo:
  - maximum F1 at **10** epochs of training

  - **1%** of the training set has the same F1 as the baseline model with **10%** of the training set

UNIVERSAL SENTENCE ENCODER

What?

- encoding sentences into embedding vectors
  -> facing NLP transfer tasks with sentence embeddings

How?

- two variants of encoding models
  (trade-off between accuracy & computational effort)

  - transformer architecture
  - deep averaging network (DAN)

TRANSFORMER

TARGETING HIGH ACCURACY!

- designed for a general purpose

- multi-task learning:

- a Skip-Thought like task

- an input-response task

- a classification task (to train on supervised data)

Skip-Thought replaces LSTM!

DEEP AVERAGING NETWORK (DAN)

TARGETING EFFICIENT COMPUTE RESOURCES!

- Averaging of word embeddings and bi-gram embeddings
- afterwards passed through a feedforward deep neural network

multi-task learning again:

one single DAN encoder is used for all 3 tasks!

- Advantage to Transformer:
  compute time is linear in the length of the input sentence!

IS TECHNOLOGY BIASED?

Target words	Attrib. words	Ref	GloVe		Uni. Enc. (DAN)	
			d	p	d	p
Eur.-American vs Afr.-American names	Pleasant vs. Unpleasant 1	*a*	1.41	10^{-8}	0.361	0.035
Eur.-American vs. Afr.-American names	Pleasant vs. Unpleasant from (a)	*b*	1.50	10^{-4}	-0.372	0.87
Eur.-American vs. Afr.-American names	Pleasant vs. Unpleasant from (*c*)	*b*	1.28	10^{-3}	0.721	0.015
Male vs. female names	Career vs family	*c*	1.81	10^{-3}	0.0248	0.48
Math vs. arts	Male vs. female terms	*c*	1.06	0.018	0.588	0.12
Science vs. arts	Male vs female terms	*d*	1.24	10^{-2}	0.236	0.32
Mental vs. physical disease	Temporary vs permanent	*e*	1.38	10^{-2}	1.60	0.0027
Young vs old peoples names	Pleasant vs unpleasant	*c*	1.21	10^{-2}	1.01	0.022
Flowers vs. insects	Pleasant vs. Unpleasant	*a*	1.50	10^{-7}	1.38	10^{-7}
Instruments vs. Weapons	Pleasant vs Unpleasant	*a*	1.53	10^{-7}	1.44	10^{-7}

Table 4: Word Embedding Association Tests (WEAT) for GloVe and the Universal Encoder. Effect size is reported as Cohen's d over the mean cosine similarity scores across grouped attribute words. Statistical significance is reported for 1 tailed p-scores. The letters in the *Ref* column indicates the source of the IAT word lists: (*a*) Greenwald et al. (1998) (*b*) Bertrand and Mullainathan (2004) (*c*) Nosek et al. (2002a) (*d*) Nosek et al. (2002b) (*e*) Monteith and Pettit (2011).

RESOURCE USAGE

	Computation Time	Memory Usage
Transformer	$O(n^2)$	$O(n)$
DAN	$O(n^2)$	$O(1)$

RESULTS

- In general: Transformer performs as good or better than DAN
- for some tasks: DAN performs as good or better than Transformer

best performance on <u>most tasks</u>:
models that make use of both sentence and word level transfer!

- transfer learning is most helpful if less training data is available
- for each NLP task the trade-off between accuracy & complexity (computation & memory) has to be considered!