

Improving methods to learn word representations for efficient semantic similarities computations

Julien Tissier

PhD defense – May 4th 2020

Université de Saint-Etienne

Jury members:

Massih-Reza AMINI	Professeur	Université Grenoble Alpes	Rapporteur
Julien VELCIN	Professeur	Université de Lyon 2	Rapporteur
Elisa FROMONT	Professeure	Université de Rennes 1	Examinatrice
Laure SOULIER	Mâitre de Conférences	Sorbonne Université	Examinatrice
Christophe GRAVIER	Mâitre de Conférences	Université de Saint-Etienne	Directeur de thèse
Amaury HABRARD	Professeur	Université de Saint-Etienne	Co-directeur de thèse



UMR • CNRS • 5516 • SAINT-ÉTIENNE

Improving methods to learn word representations for efficient semantic similarities computations

"Hey Siri"



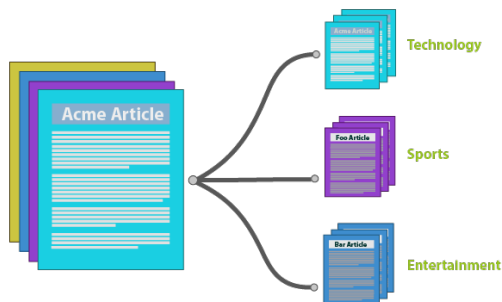
"OK Google"



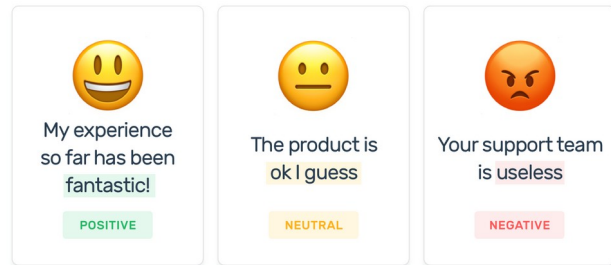
Virtual assistant



Automatic translation

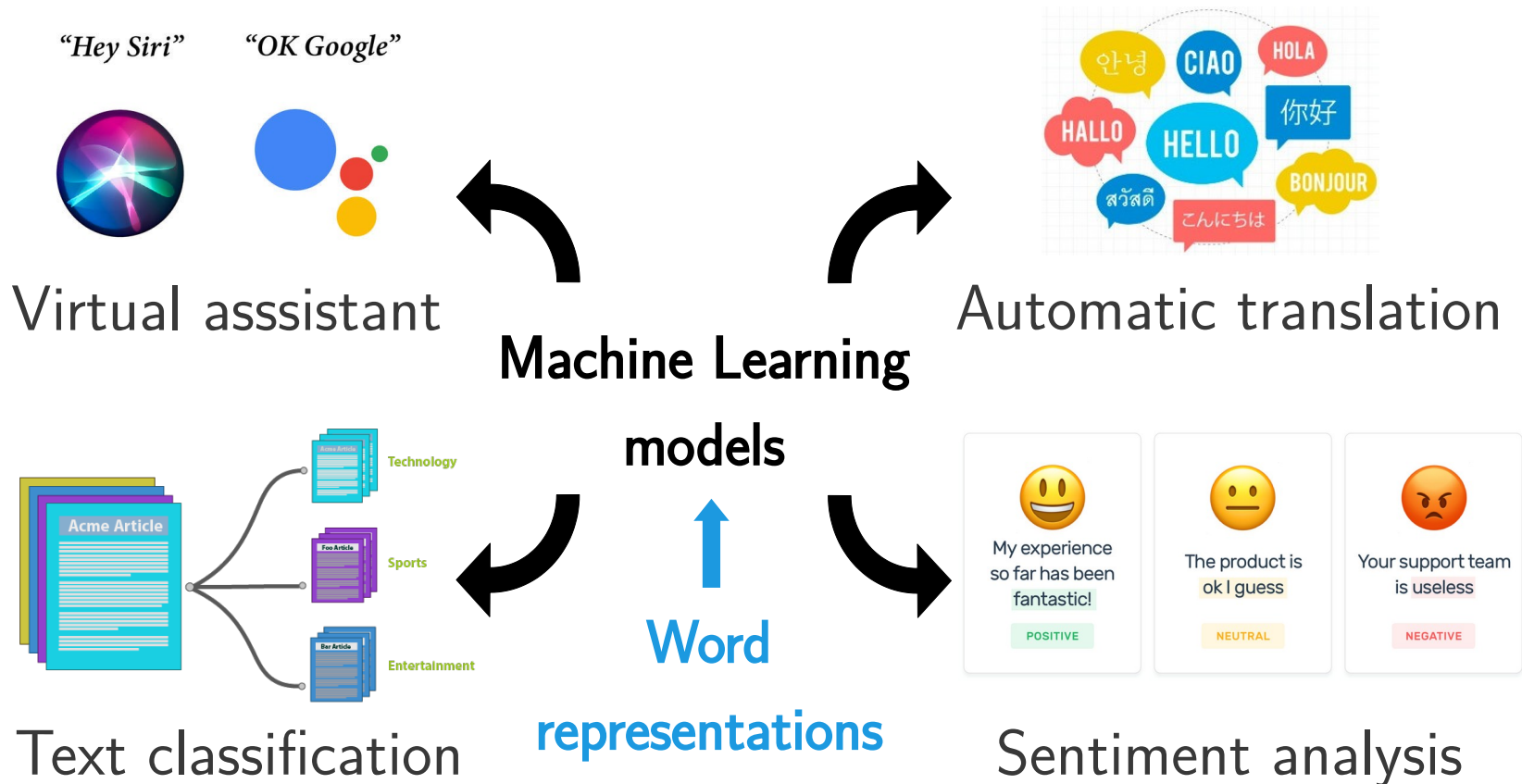


Text classification

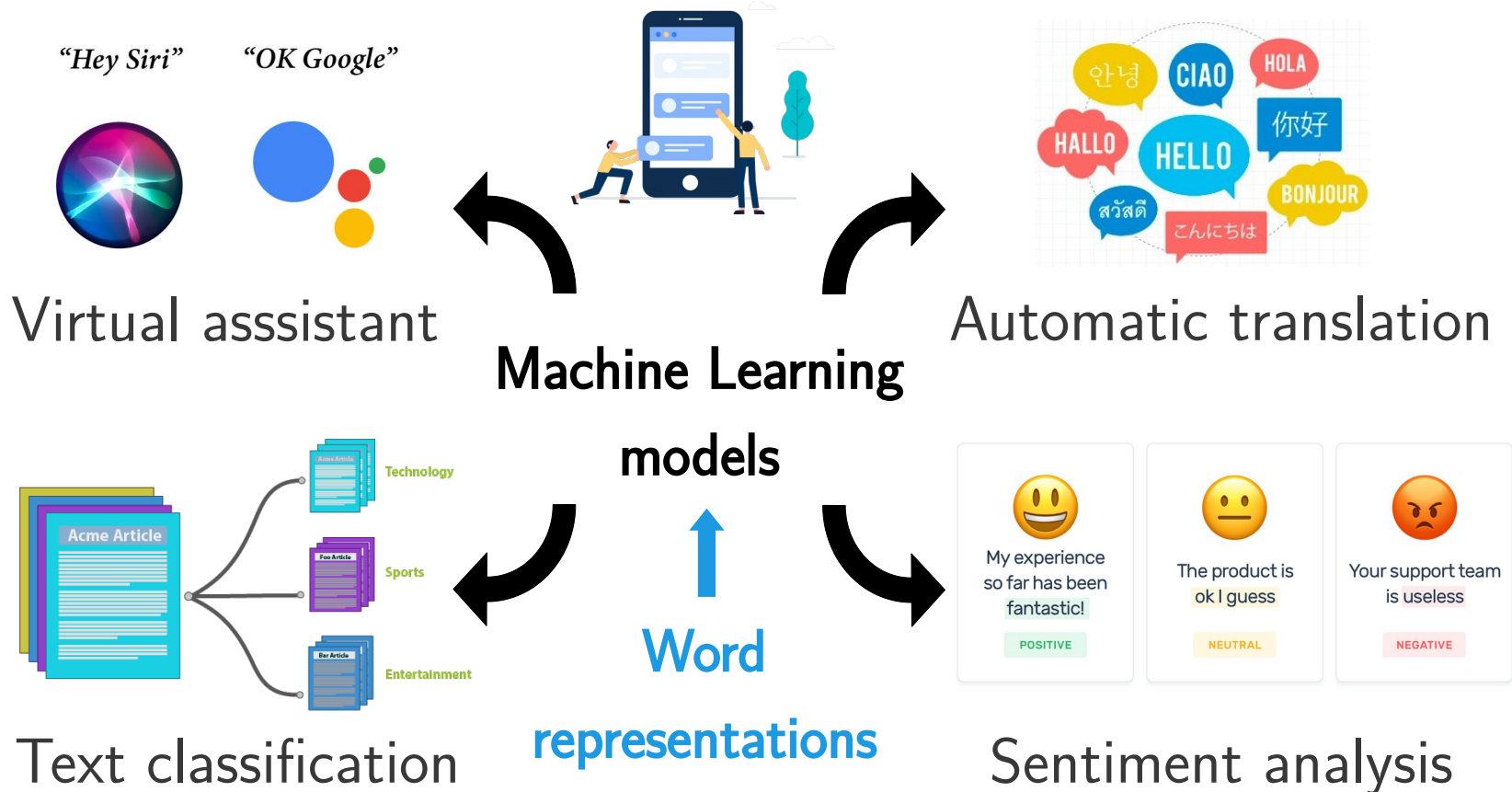


Sentiment analysis

Improving methods to learn word representations for efficient semantic similarities computations



Improving methods to learn word representations for efficient semantic similarities computations



Contents

1. Introduction

Liverpool →	-0.6	0.1
Tottenham →	-0.4	0.6
goal →	-0.2	-0.1

3. Binary Word Embeddings

Liverpool →	0	1	1	0
Tottenham →	0	0	1	0
goal →	0	1	0	0

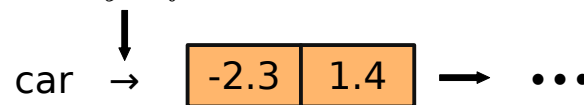
2. Word Embeddings and Dictionaries

“car: a road vehicle, typically with four wheels”

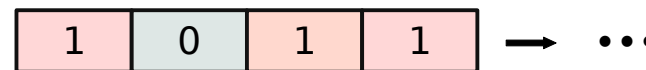


4. Conclusion

Dictionary definitions



Binarization



Contents

1. Introduction

Liverpool →	-0.6	0.1
Tottenham →	-0.4	0.6
goal →	-0.2	-0.1

2. Word Embeddings and Dictionaries

“car: a road vehicle, typically with four wheels”

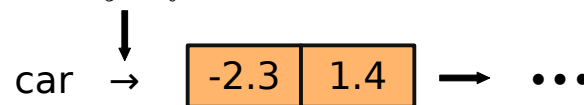


3. Binary Word Embeddings

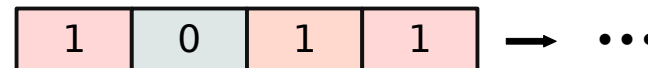
Liverpool →	0	1	1	0
Tottenham →	0	0	1	0
goal →	0	1	0	0

4. Conclusion

Dictionary definitions

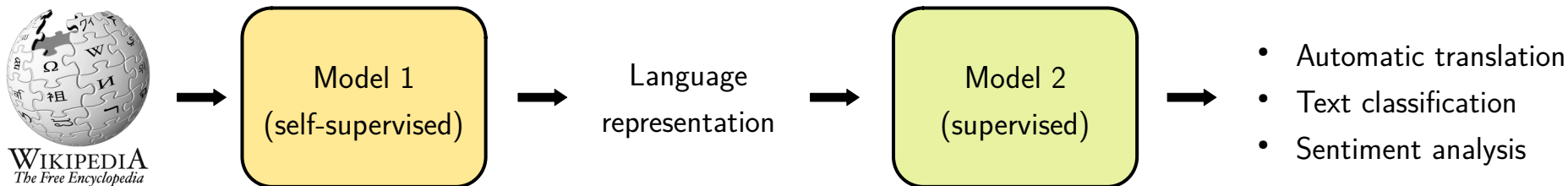


Binarization



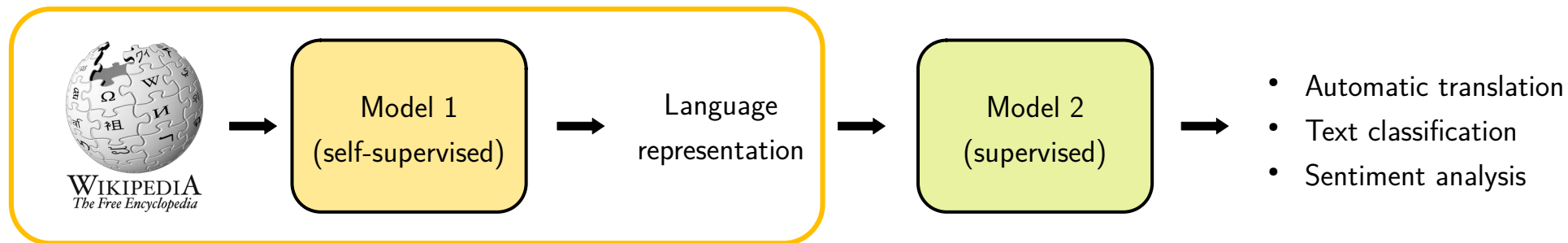
Classic NLP pipeline

- Use 2 models: 1 to learn the representations, 1 to solve the task
- Representation of the language have to contain linguistic information, so the downstream model can have access to knowledge of the language
- Usually learned by self-supervised models from large corpora



Classic NLP pipeline

- Use 2 models: 1 to learn the representations, 1 to solve the task
- Representation of the language have to contain linguistic information, so the downstream model can have access to knowledge of the language
- Usually learned by self-supervised models from large corpora

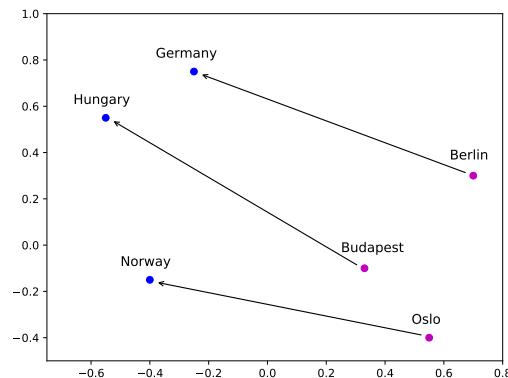


Context of this PhD

Word embeddings

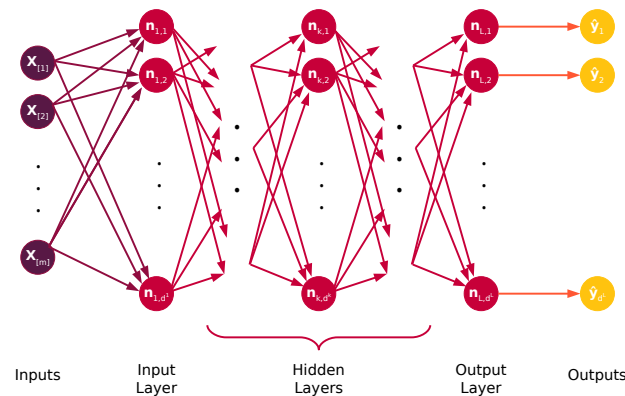
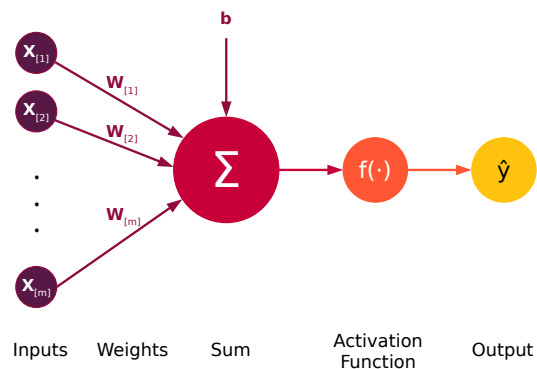
- Representation of words of a language
- Word embeddings: associating one numerical vector to each word of a language
- These vectors should reflect the linguistic information of words
 - Semantic (meaning of words)
 - Syntactic (properties of words)

Berlin →	0.7	0.3
Budapest →	0.3	-0.1
Oslo →	0.55	-0.4



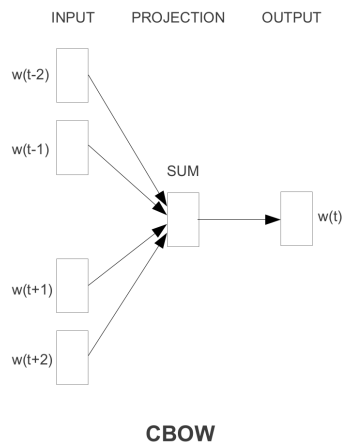
Learning word embeddings: neural network

- Machine learning models composed of layers of neurons
- Given an input, compute an output. Trained to minimize a loss based on the computed output and the expected output
- Trained with gradient descent (and backpropagation)



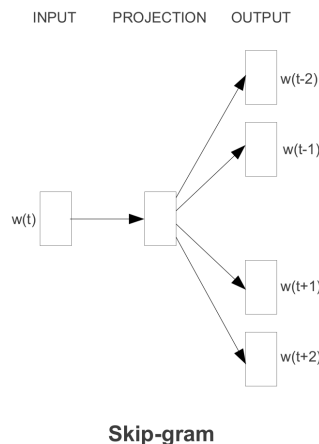
Learning word embeddings: neural network

- Shallow

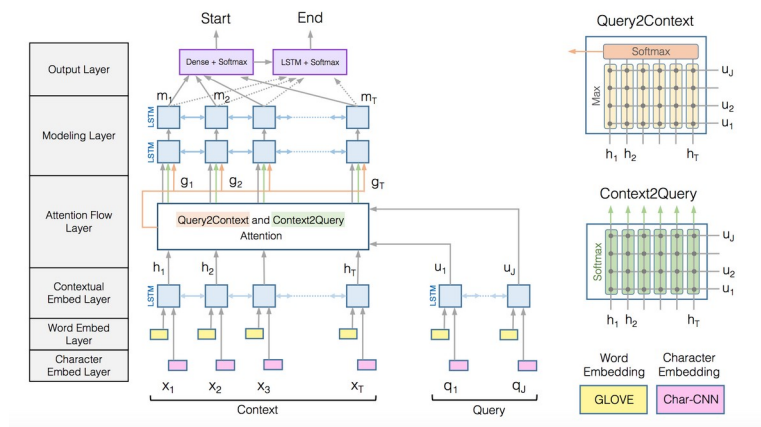


Architecture of word2vec

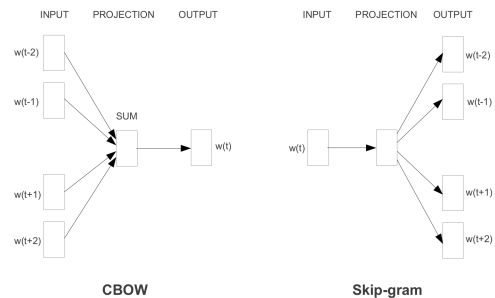
- Deep



Architecture of BERT



Learning word embeddings



Quantity of linguistic information
used during training

BERT, 2019

ELMo, 2018

Sense embeddings, 2015

Retrofitting, 2015

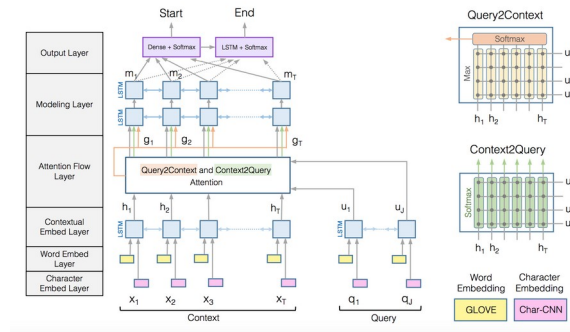
Yu & Dredze, 2014

word2vec 2013,
GloVe 2014

fasttext, 2017

Collobert et al., 2011

Collobert & Weston, 2008



Small models

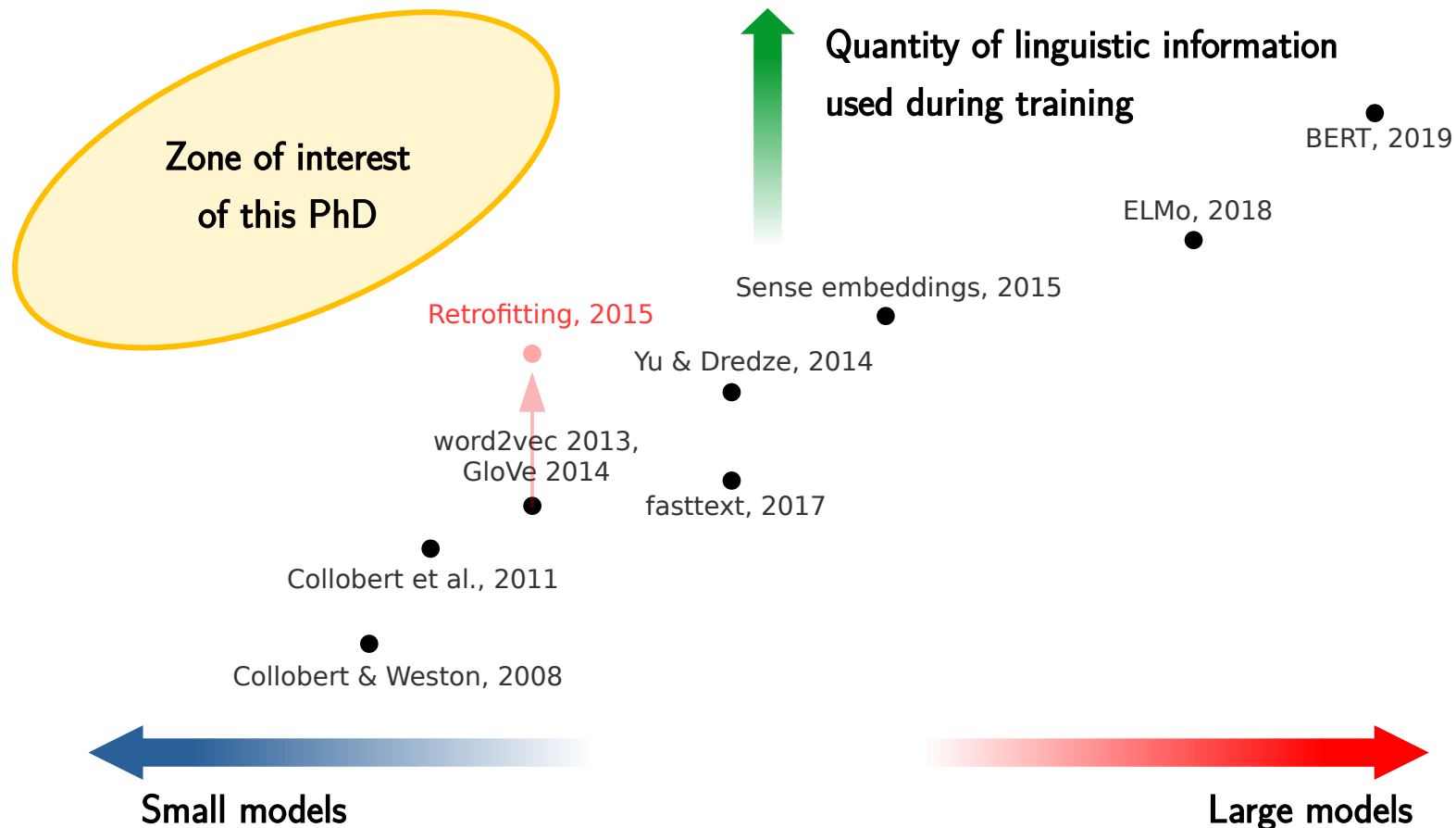


Large models

Learning word embeddings

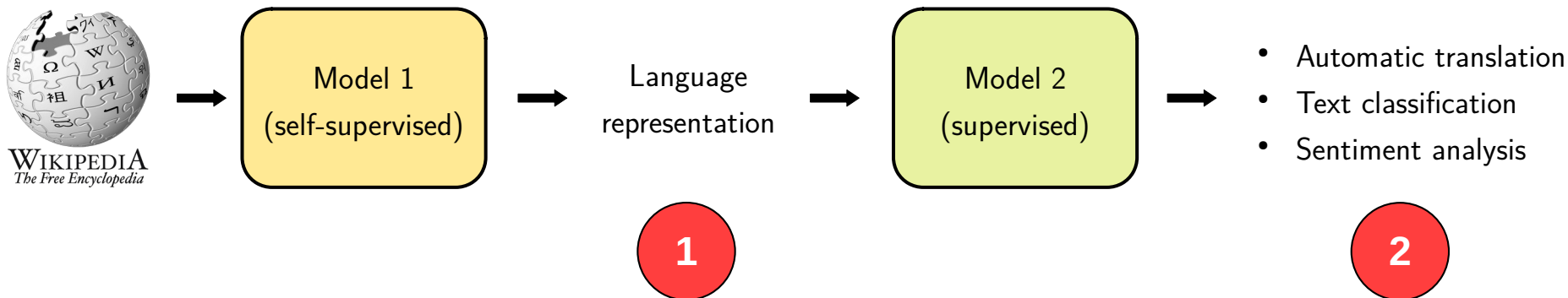
- Recent models are able to use more language information during training ...
- ... but this implies larger model architectures
- More and more downstream tasks are run on mobile devices, where large models and word embeddings cannot be deployed
- How to learn word embeddings with large quantity of information but small model architecture?

Learning word embeddings



Evaluating word embeddings

- 1) Quality of captured information → **intrinsic evaluation**
- 2) Useful in downstream task → **extrinsic evaluation**



Contents

1. Introduction

Liverpool →	-0.6	0.1
Tottenham →	-0.4	0.6
goal →	-0.2	-0.1

2. Word Embeddings and Dictionaries

“car: a road vehicle, typically with four wheels”

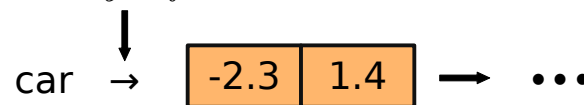


3. Binary Word Embeddings

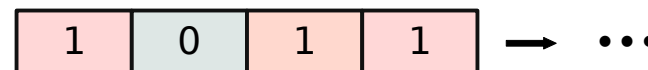
Liverpool →	0	1	1	0
Tottenham →	0	0	1	0
goal →	0	1	0	0

4. Conclusion

Dictionary definitions



Binarization



Classic approach to learn word embeddings

- Use co-occurrence of words in large text corpora

word2vec (2013)
GloVe (2014)
fasttext (2017)

<div data-bbox="270 516 683 569"> <div data-bbox="270 516 463 569">The</div> <div data-bbox="463 516 570 569">quick</div> <div data-bbox="570 516 683 569">brown</div> </div> <div data-bbox="683 516 1334 569">fox jumps over the lazy dog. →</div>	<div data-bbox="683 516 1334 569"> <div data-bbox="683 516 1334 569">(the, quick)</div> <div data-bbox="683 569 1334 569">(the, brown)</div> </div>
<div data-bbox="270 569 683 622"> <div data-bbox="270 569 463 622">The</div> <div data-bbox="463 569 570 622">quick</div> <div data-bbox="570 569 683 622">brown</div> <div data-bbox="683 569 749 622">fox</div> </div> <div data-bbox="749 569 1334 622">jumps over the lazy dog. →</div>	<div data-bbox="683 569 1334 622"> <div data-bbox="683 569 1334 622">(quick, the)</div> <div data-bbox="683 622 1334 622">(quick, brown)</div> <div data-bbox="683 622 1334 622">(quick, fox)</div> </div>
<div data-bbox="270 622 683 675"> <div data-bbox="270 622 463 675">The</div> <div data-bbox="463 622 570 675">quick</div> <div data-bbox="570 622 683 675">brown</div> <div data-bbox="683 622 749 675">fox</div> <div data-bbox="749 622 855 675">jumps</div> </div> <div data-bbox="855 622 1334 675">over the lazy dog. →</div>	<div data-bbox="683 622 1334 675"> <div data-bbox="683 622 1334 675">(brown, the)</div> <div data-bbox="683 675 1334 675">(brown, quick)</div> <div data-bbox="683 675 1334 675">(brown, fox)</div> <div data-bbox="683 675 1334 675">(brown, jumps)</div> </div>
<div data-bbox="270 675 683 728"> <div data-bbox="270 675 463 728">The</div> <div data-bbox="463 675 570 728">quick</div> <div data-bbox="570 675 683 728">brown</div> <div data-bbox="683 675 749 728">fox</div> <div data-bbox="749 675 855 728">jumps</div> <div data-bbox="855 675 949 728">over</div> </div> <div data-bbox="949 675 1334 728">the lazy dog. →</div>	<div data-bbox="683 675 1334 728"> <div data-bbox="683 675 1334 728">(fox, quick)</div> <div data-bbox="683 728 1334 728">(fox, brown)</div> <div data-bbox="683 728 1334 728">(fox, jumps)</div> <div data-bbox="683 728 1334 728">(fox, over)</div> </div>

Classic approach to learn word embeddings

- Use co-occurrence of words to move closer the embeddings of related words (e.g. fox/brown)
- $$\text{Loss}(\mathbf{v}'_{\text{brown}}, \mathbf{v}_{\text{fox}}) = \underbrace{-\log(\sigma(\mathbf{v}'_{\text{brown}}^\top \mathbf{v}_{\text{fox}}))}_{\text{Move closer the vectors of cooccurring words}} - \underbrace{\sum_{i=1}^k \mathbb{E}_{w_i \sim P_{w_{\text{fox}}}} \log(\sigma(-\mathbf{v}'_i^\top \mathbf{v}_{\text{fox}}))}_{\text{Move further the vectors of random words (negative sampling)}}$$
- $$\text{Global loss} = \frac{1}{T} \sum_{t=1}^T \sum_{\substack{-n \leq j \leq n \\ j \neq 0}} \text{Loss}(\mathbf{v}'_{t+j}, \mathbf{v}_t)$$

Classic approach to learn word embeddings

- Words within the same window are not always related
- Semantic relations like synonymy happen rarely within the same window → not properly captured by this approach

Classic approach to learn word embeddings

- Words within the same window are not always related
- Semantic relations like synonymy happen rarely within the same window
→ not properly captured by this approach
- **Solution: add information contained in dictionaries**
 - Strong semantic information
 - Weak supervision
 - Move closer related words

dict2vec: Learning word embeddings using lexical dictionaries

- Use dictionary definitions to generate pairs of related words
 - **car**: a road **vehicle**, typically with four **wheels**, powered by an internal combustion engine and able to carry a small number of people.
 - **vehicle**: a thing used for transporting people or goods, especially on land, such as a **car**, lorry or cart.
 - **wheel**: a circular object that revolves on an axle and is fixed below a **vehicle** or other object to enable it to move easily over the ground.

dict2vec: Learning word embeddings using lexical dictionaries

- Use dictionary definitions to generate pairs of related words
 - **car**: a road **vehicle**, typically with four **wheels**, powered by an internal combustion engine and able to carry a small number of people.
 - **vehicle**: a thing used for transporting people or goods, especially on land, such as a **car**, lorry or cart.
 - **wheel**: a circular object that revolves on an axle and is fixed below a **vehicle** or other object to enable it to move easily over the ground.
- **Strong pair**: mutual inclusion (**car** – **vehicle**)

dict2vec: Learning word embeddings using lexical dictionaries

- Use dictionary definitions to generate pairs of related words
 - **car**: a road **vehicle**, typically with four **wheels**, powered by an internal combustion engine and able to carry a small number of people.
 - **vehicle**: a thing used for transporting people or goods, especially on land, such as a **car**, lorry or cart.
 - **wheel**: a circular object that revolves on an axle and is fixed below a **vehicle** or other object to enable it to move easily over the ground.
- **Strong pair**: mutual inclusion (**car** – **vehicle**)
- **Weak pair**: one-sided inclusion (**car** – **wheel**), (**wheel** – **vehicle**)

dict2vec: Learning word embeddings using lexical dictionaries

- Loss $(\mathbf{v}'_c, \mathbf{v}_t) = -\log(\sigma(\mathbf{v}'_c^\top \mathbf{v}_t))$ // Move closer the vectors of co-occurring words
// Positive sampling
$$- \beta_s \sum_{w_i \in S_{strong}(w_c)} \log(\sigma(\mathbf{v}'_i^\top \mathbf{v}_c)) - \beta_w \sum_{w_j \in W_{weak}(w_c)} \log(\sigma(\mathbf{v}'_j^\top \mathbf{v}_c))$$

// Controlled negative sampling
$$- \sum_{\substack{w_i \in R_{random}(w_t) \\ w_i \notin S_{strong}(w_t) \\ w_i \notin W_{weak}(w_t)}} \log(\sigma(-\mathbf{v}'_i^\top \mathbf{v}_t))$$
- Global loss
$$= \frac{1}{T} \sum_{t=1}^T \sum_{\substack{-n \leq j \leq n \\ j \neq 0}} \text{Loss}(\mathbf{v}'_t, \mathbf{v}_{t+j})$$

dict2vec: Learning word embeddings using lexical dictionaries

- Trained on the full Wikipedia corpus (July 2017) with pairs generated from 4 dictionaries
- Evaluation
 - Intrinsic (word semantic similarity)
 - Extrinsic (document classification, sentiment analysis)

Results: semantic similarity evaluation

- Datasets composed of pairs of words rated by humans
- Compared to **word2vec** and **fasttext**

television - radio: 6.77
bank - money: 8.12
opera - industry: 2.63

	Wikipedia		
	word2vec	fasttext	dict2vec
MC-30	0.809	0.831	0.860
MEN-TR-3k	0.733	0.752	0.756
MTurk-287	0.660	0.672	0.661
MTurk-771	0.623	0.631	0.696
RG-65	0.787	0.817	0.875
RW	0.407	0.464	0.482
SimVerb	0.186	0.222	0.384
WS353-ALL	0.705	0.729	0.756
WS353-REL	0.664	0.687	0.702
WS353-SIM	0.757	0.775	0.781
YP-130	0.502	0.533	0.646
<i>W.Average</i>	0.476	0.508	0.573

Results: semantic similarity evaluation

- Not fair because **dict2vec** uses additional knowledge
- Also add the definition into the training corpus

	Wikipedia			Wikipedia + definitions		
	word2vec	fasttext	dict2vec	word2vec	fasttext	dict2vec
MC-30	0.809	0.831	0.860	0.826	0.815	0.847
MEN-TR-3k	0.733	0.752	0.756	0.728	0.751	0.755
MTurk-287	0.660	0.672	0.661	0.656	0.671	0.660
MTurk-771	0.623	0.631	0.696	0.620	0.638	0.694
RG-65	0.787	0.817	0.875	0.802	0.820	0.867
RW	0.407	0.464	0.482	0.427	0.468	0.476
SimVerb	0.186	0.222	0.384	0.214	0.233	0.379
WS353-ALL	0.705	0.729	0.756	0.721	0.723	0.758
WS353-REL	0.664	0.687	0.702	0.681	0.686	0.703
WS353-SIM	0.757	0.775	0.781	0.767	0.779	0.781
YP-130	0.502	0.533	0.646	0.475	0.553	0.607
<i>W.Average</i>	0.476	0.508	0.573	0.488	0.512	0.570

Results: text classification evaluation

- Document classification (AG-News, DBPedia), Sentiment analysis (Yelp)
- On par or slightly better → not over-specialized for semantic similarities

	Wikipedia			Wikipedia + definitions		
	word2vec	fasttext	dict2vec	word2vec	fasttext	dict2vec
AG-News	88.5	88.7	88.1	88.5	88.7	88.4
DBPedia	96.6	96.7	96.8	96.6	96.7	96.9
Yelp Pol.	86.5	87.2	87.6	86.7	87.4	87.5
Yelp Full	50.6	51.2	51.6	50.6	51.4	51.8

Results: dictionaries vs. WordNet

- WordNet is a knowledge base of related words
- Use pairs of WordNet related words in the dict2vec model
- Using dictionary pairs is better than using WordNet pairs

	50M	200M	full
No pairs	0.453	0.471	0.488
With WordNet pairs	0.564	0.566	0.559
With dictionary pairs	0.569	0.569	0.571


Summary

- **dict2vec** is a new model that incorporates additional information during training of word embeddings
- Use dictionaries to generate pairs of related words
- **dict2vec** introduces the positive sampling to incorporate these pairs of related words
- Large improvements of semantic information captured in word embeddings

 [tca19 / dict2vec](#)

 Watch

3

 Star

82

 Fork

19

Contents

1. Introduction

Liverpool →	-0.6	0.1
Tottenham →	-0.4	0.6
goal →	-0.2	-0.1

3. Binary Word Embeddings

Liverpool →	0	1	1	0
Tottenham →	0	0	1	0
goal →	0	1	0	0

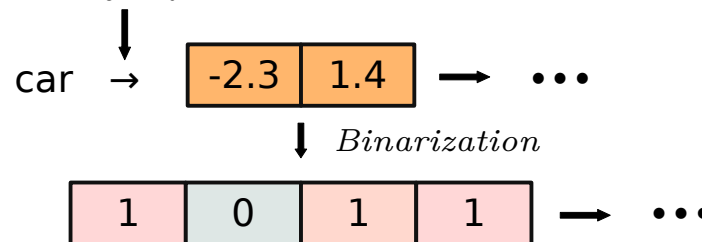
2. Word Embeddings and Dictionaries

“car: a road vehicle, typically with four wheels”



4. Conclusion

Dictionary definitions



Classic real-valued word embeddings

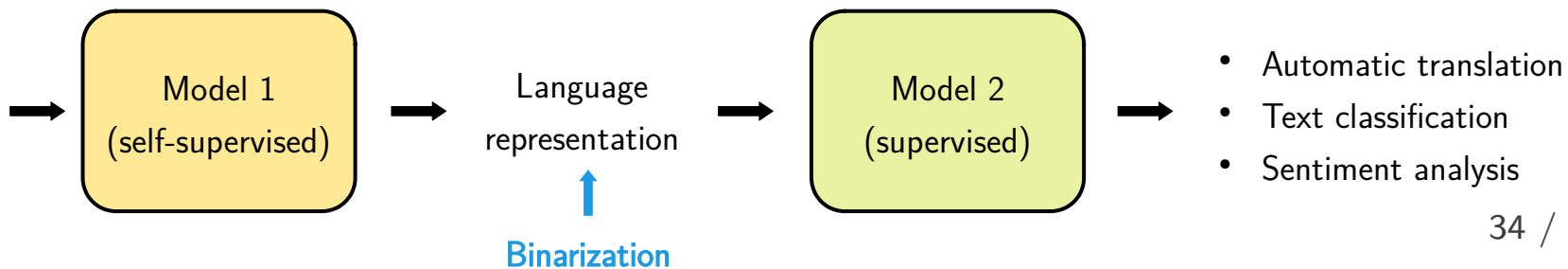
- Require a lot of space for storage
 - Millions of words in vocabulary
 - Usually 300 dimensions per vector
 - A real value (float) requires 32 bits for storage
 - For 1M vectors, 1.2 gigabytes are required
- Not suitable for embedded devices
 - They have limited memory (both RAM and disk)
 - Low computing power for float operations

Binary word embeddings

- Solution: use binary word embeddings, *i.e.* associate a m -bits vector to each word of a vocabulary
- Faster vector operations
 - Cosine similarity requires $O(n)$ additions and multiplications
 - Binary similarity requires a **XOR** and a **popcount()** operations
- Small memory size
 - 9600 bits per real-valued vector (300 dimensions, 32 bits per value)
 - 128 or 256 bits per binary vector
 - 16.1 MB to store 1M vectors of 128 bits (vs. 1.2 GB)
 - Computations can be done locally; no need of sending data to computing servers

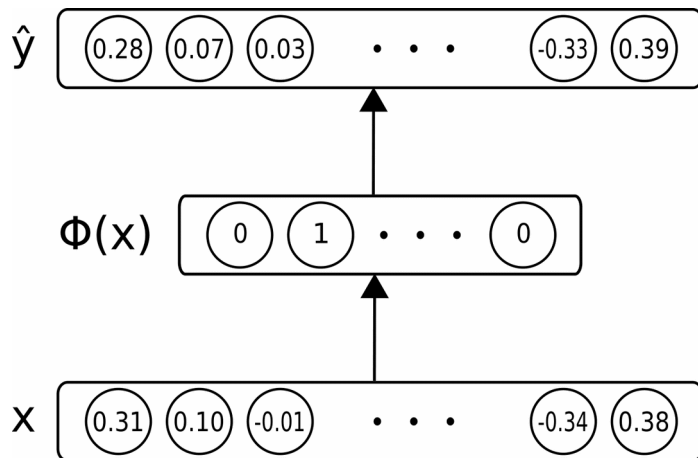
Binary word embeddings

- Binarize instead of training binary vectors
 - Many pre-trained vectors already available. Some are specific (subwords, dictionaries...).
 - Need a general method that works for all types of word embeddings
- Why not a naive binarization?
 - To benefit from hardware optimizations for computations, binary word vectors need to have a size in adequacy with CPU register size (64, 128 or 256 bits)
 - Pre-trained vectors have usually 300 dimensions (so naive binary vectors would have 300 bits)



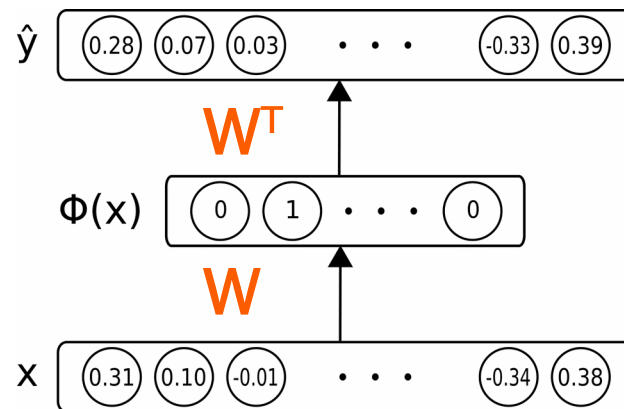
NLB: Near-lossless binarization of word embeddings

- Use pre-trained vectors as input ($\mathbf{x} \in \mathbb{R}^m$)
- Latent representation $\Phi(\mathbf{x})$ is binary ($\Phi(\mathbf{x}) \in \mathbb{B}^n$)
- Reconstruct a real-valued vector $\hat{\mathbf{y}} \in \mathbb{R}^m$ given $\Phi(\mathbf{x})$
- Trained to minimize the reconstruction loss: $\text{Loss}(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{y}}\|^2$



NLB: Near-lossless binarization of word embeddings

- Let $\mathbf{W} \in \mathbb{R}^{n \times m}$, $\mathbf{c} \in \mathbb{R}^m$ and $\mathbf{x}_i \in \mathbb{R}^m$
- Encoding
 - Heaviside function $h(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$
 - $\mathbf{b}_i = \Phi(\mathbf{x}_i) = h(\mathbf{W} \cdot \mathbf{x}_i^\top)$
 - Can construct binary vectors of arbitrary size
- Decoding
 - $\hat{\mathbf{y}}_i = \tanh(\mathbf{W}^\top \cdot \Phi(\mathbf{x}_i) + \mathbf{c})$



NLB: Near-lossless binarization of word embeddings

- Reconstruction loss: minimize the distance between \mathbf{x}_i and $\hat{\mathbf{y}}_i$

$$\ell_{rec}(\mathbf{x}_i) = \frac{1}{d} \|\mathbf{x}_i - \hat{\mathbf{y}}_i\|^2 = \frac{1}{d} \sum_{j=1}^d (\mathbf{x}_{i[j]} - \hat{\mathbf{y}}_{i[j]})^2$$

- Regularization loss: minimize the correlation between each binary dimension

$$\ell_{reg} = \frac{1}{2} \left\| \mathbf{W}^\top \mathbf{W} - \mathbf{I} \right\|^2$$

- Global objective function

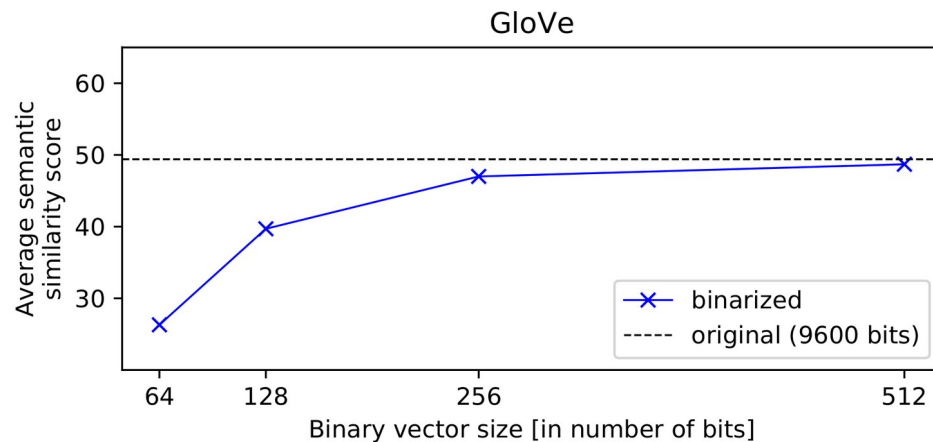
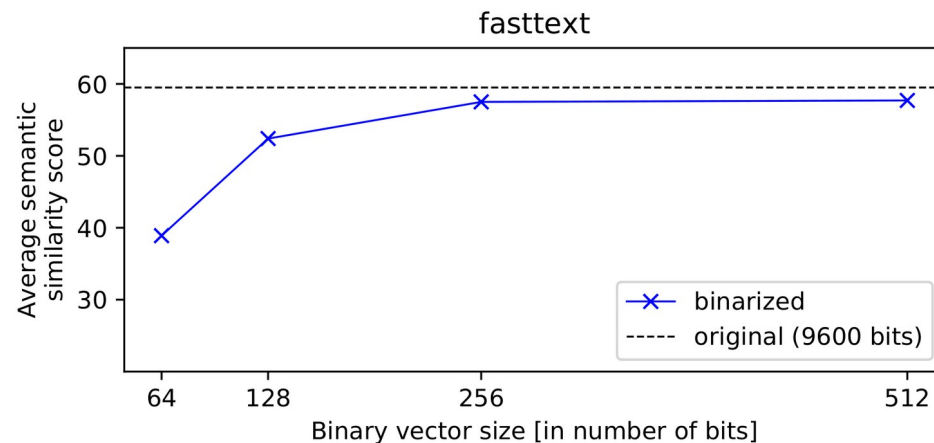
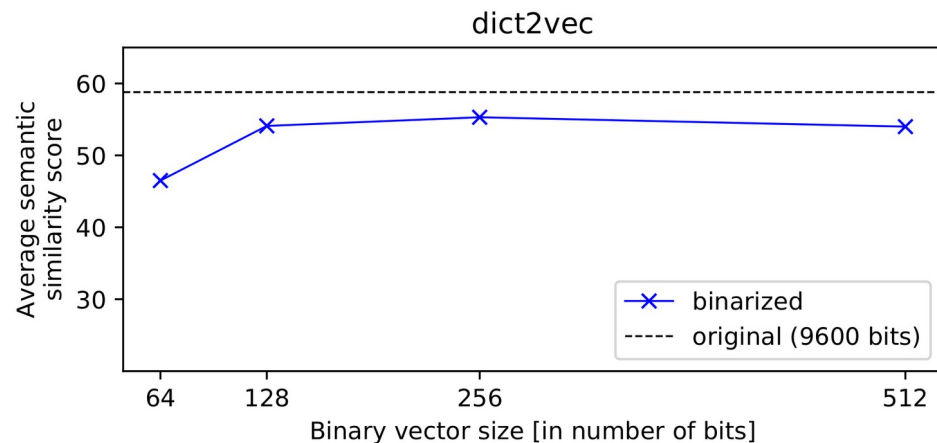
$$\mathcal{L} = \sum_{\mathbf{x}_i \in \mathbf{M}} \ell_{rec}(\mathbf{x}_i) + \lambda_{reg} \ell_{reg}$$

Results: semantic similarity evaluation

- Binary vector similarity: $\text{sim}(\mathbf{b}_1, \mathbf{b}_2) = \frac{n_{11} + n_{00}}{n}$
- Binarized **dict2vec** (2.3M), **fasttext** (1M) and **GloVe** (400k)
- Best average results achieved with 512 bits (18 times smaller)

	dict2vec					fasttext					GloVe				
	\mathbb{R}	64	128	256	512	\mathbb{R}	64	128	256	512	\mathbb{R}	64	128	256	512
MEN	0.746					0.807					0.737				
NLB	-	0.661	0.713	0.703	0.713	-	0.579	0.720	0.759	0.763	-	0.461	0.633	0.694	0.727
RW	0.505					0.538					0.412				
NLB	-	0.365	0.420	0.456	0.456	-	0.368	0.447	0.527	0.527	-	0.251	0.343	0.407	0.402
SimLex	0.452					0.441					0.371				
NLB	-	0.320	0.381	0.448	0.429	-	0.251	0.380	0.446	0.430	-	0.205	0.314	0.372	0.368
SimVerb	0.417					0.356					0.227				
NLB	-	0.253	0.366	0.384	0.355	-	0.192	0.267	0.337	0.351	-	0.078	0.187	0.229	0.230
WordSim	0.725					0.697					0.609				
NLB	-	0.637	0.716	0.696	0.666	-	0.503	0.691	0.700	0.703	-	0.301	0.449	0.566	0.603
<i>W.average</i>	0.588					0.595					0.494				
NLB	-	0.465	0.541	0.553	0.540	-	0.389	0.524	0.575	0.577	-	0.263	0.397	0.470	0.487

Results: semantic similarity evaluation



Results: text classification evaluation

- Use the binary vectors to initialize the weights of a downstream classification model
- Binary vectors are on par or slightly outperform real-valued original vectors

	dict2vec					fasttext					GloVe				
	\mathbb{R}	64	128	256	512	\mathbb{R}	64	128	256	512	\mathbb{R}	64	128	256	512
AG-News	89.0					86.9					89.5				
NLB	-	85.3	85.9	87.7	87.8	-	84.5	85.9	87.3	87.7	-	84.0	87.2	88.5	88.5
LSH	-	78.8	82.6	86.1	88.1	-	77.5	83.3	86.1	88.8	-	83.5	86.6	88.4	88.6
DBpedia	97.6					95.0					97.2				
NLB	-	94.1	96.1	97.0	97.3	-	91.7	95.1	96.6	97.3	-	90.9	95.0	96.8	97.2
LSH	-	89.6	94.2	96.5	97.4	-	87.4	93.8	96.2	97.2	-	90.4	94.2	96.3	97.2
Yahoo Ans.	68.1					67.2					68.1				
NLB	-	60.7	63.8	66.0	66.8	-	60.4	63.9	66.4	67.8	-	57.5	62.5	66.4	66.1
LSH	-	52.3	59.9	64.5	67.1	-	52.2	59.5	64.9	66.9	-	56.8	62.0	65.3	67.0
Amazon Full	47.5					49.0					47.1				
NLB	-	39.9	43.9	46.8	47.7	-	39.0	43.9	47.9	49.8	-	37.4	42.6	46.7	47.8
LSH	-	38.3	42.5	45.6	48.1	-	38.6	42.7	47.3	49.5	-	37.9	43.0	45.9	48.6

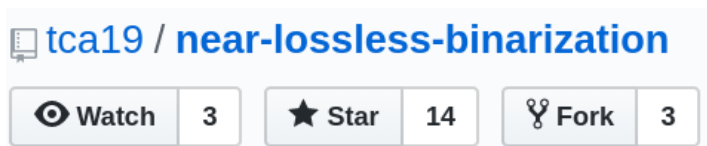
Results: Top-K queries speed improvements

- Measure the time needed to find the K vectors with the highest similarity given a query vector, for both binary and original real-valued vectors
- Using 256-bit vectors is 30 times faster than using real-valued vectors
- When the loading time is taken into account, they are 75 times faster

Top-K query	64-bit	128-bit	256-bit	512-bit	Real-valued
Top 1	2.71	2.87	3.23	4.28	97.89
Top 10	2.72	2.89	3.25	4.29	98.08
Top 50	2.75	2.91	3.27	4.32	98.44
Loading + Top 10	160	213	310	500	23500

Summary

- Simple architecture to binarize any real-valued word embeddings
- Reduce the vector size by 37.5 with only a 2% performance loss
- Perform a top-K query 30 times faster than with real-valued vectors



Contents

1. Introduction

Liverpool →	-0.6	0.1
Tottenham →	-0.4	0.6
goal →	-0.2	-0.1

3. Binary Word Embeddings

Liverpool →	0	1	1	0
Tottenham →	0	0	1	0
goal →	0	1	0	0

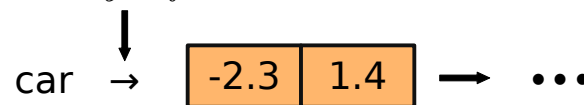
2. Word Embeddings and Dictionaries

“car: a road vehicle, typically with four wheels”

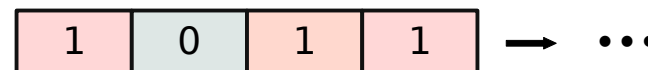


4. Conclusion

Dictionary definitions



Binarization



Conclusion

- Presented two novel complementary methods to learn word embeddings
 - **dict2vec** [1] uses additional information from lexical dictionaries to improve the semantic knowledge captured in word embeddings

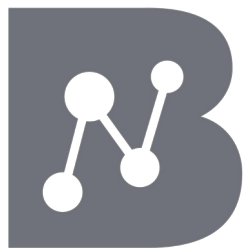
[1] Julien Tissier, Christophe Gravier, and Amaury Habrard. “Dict2vec: Learning word embeddings using lexical dictionaries”. In *Conference Methods in Natural Language Processing (EMNLP 2017)*, pages 254–263, 2017.

- **NLB** [2] transforms pre-trained vectors to binary word embeddings, resulting in smaller memory size and allowing faster semantic similarity computations on word vectors

[2] Julien Tissier, Christophe Gravier, and Amaury Habrard. “Near-lossless binarization of word embeddings”. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019)*, volume 33, pages 7104–7111, 2019.

Perspectives

- Polysemy information has been disregarded in **dict2vec**
- Provides useful information for word sense disambiguation systems
- Use the multi-sense information to learn disambiguated word embeddings



BabelNet



SAPIENZA
UNIVERSITÀ DI ROMA

Perspectives

- Binarization of representations can be extended to sentence or document representations
- Binarization can also be applied to the hidden representations of downstream models
- Proposal to Facebook Research grant award

Perspectives

- Learning word embeddings of different languages (not only English)
- Classification of Wikipedia pages among 200 hierarchical classes and 30 languages



Thank you for your attention