

# Finding Word Sense Embeddings of Known Meaning

A method for refitting word sense embeddings, using  
a single example, by application of Bayes' theorem to  
the language model

**Lyndon White,**  
Roberto Togneri, Wei Liu, Mohammed Bennamoun

School of Electical, Electronic and Computer Engineering  
The University of Western Australia

# Words don't only have one meaning

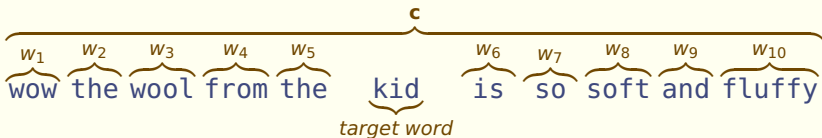
## Kid (*Noun*)

1. (a young person of either sex) "she writes books for children"; "they're just kids"; "'tiddler' is a British term for youngster"
2. (English dramatist (1558-1594))
3. (a human offspring (son or daughter) of any age) "they had three children"; "they were able to send their kids to college"
4. (young goat)

# Word embeddings represent each word as a single vector

## SkipGram Language Model:

- ▶ Input a word  $w_T$
- ▶ Output probabilities of words appearing in its context  $P(w_i | w_T)$



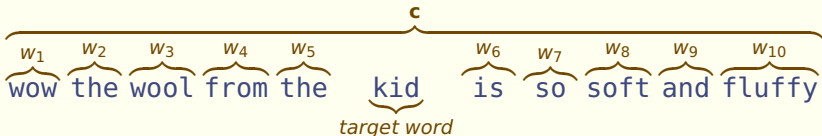
# Word embeddings represent each word as a single vector

## SkipGram Language Model:

- ▶ Input a word  $w_T$
- ▶ Output probabilities of words appearing in its context  $P(w_i | w_T)$

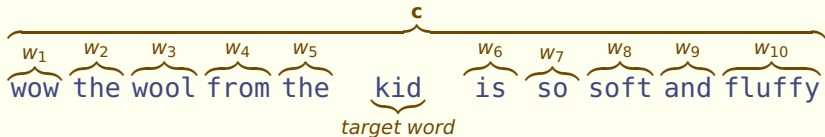
## Word Embeddings Implementation

- ▶ Represent each input word as a **vector**
- ▶ Train a neural network to estimate  $P(w_i | w_T)$
- ▶ Back-prop finds values for the input vector – i.e. good representation for the word



# Word sense embeddings represent each word as a multiple vectors

- ▶ Each word has multiple senses  $\{u_1, u_2, \dots, u_n\}$
- ▶ SkipGram Language Model becomes
  - ▶ **Input** a word sense  $u_i$
  - ▶ **Output** probabilities of words appearing in its context  $P(w_i | u_j)$



# Many sense embeddings don't produce human recognisable senses

- Embeddings are learnt by modelling what words occur near the sense

# Many sense embeddings don't produce human recognisable senses

- ▶ Embeddings are learnt by modelling what words occur near the sense
- ▶ No control over the meanings of the senses
  - ▶ Cover overlapping definitions
  - ▶ Find overly narrow meanings
  - ▶ Capture rare jargon uses

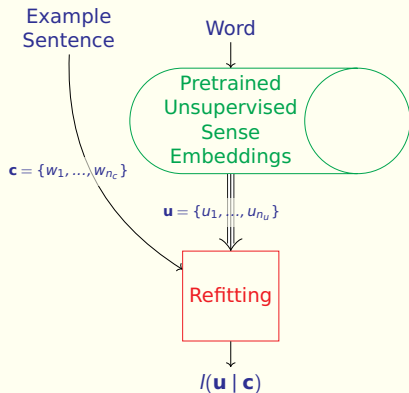
# Many sense embeddings don't produce human recognisable senses

- ▶ Embeddings are learnt by modelling what words occur near the sense
- ▶ No control over the meanings of the senses
  - ▶ Cover overlapping definitions
  - ▶ Find overly narrow meanings
  - ▶ Capture rare jargon uses
- ▶ Useful, but not interoperable with lexical knowledge bases.



We will solve this by *refitting* them to be for the sense we mean

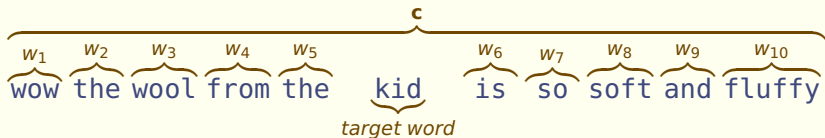
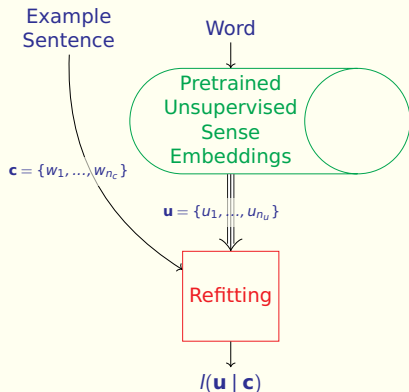
- ▶ Refitting constructs new sense embeddings out of the old.
- ▶ It uses the probabilities of example sentence occurring.
- ▶ The new embedding aligns to the meaning in that sentence.



# Refitting uses a probability weighted sum

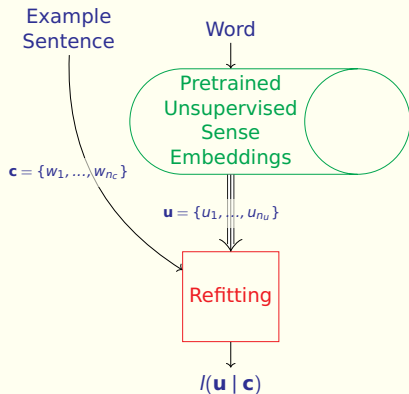
New sense embedding:

$$I(\mathbf{u} \mid \mathbf{c}) = \sum_{\forall u_i \in \mathbf{u}} u_i P(u_i \mid \mathbf{c})$$



# The probabilities are found using Bayes' theorem

Language model:  $P(w_i | u_i)$



$c$

$w_1$   $w_2$   $w_3$   $w_4$   $w_5$   $w_6$   $w_7$   $w_8$   $w_9$   $w_{10}$

wow the wool from the kid is so soft and fluffy

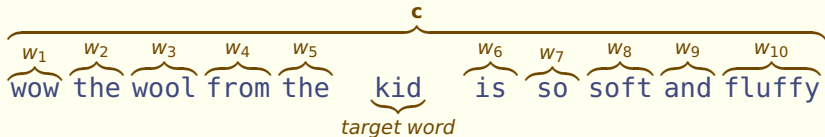
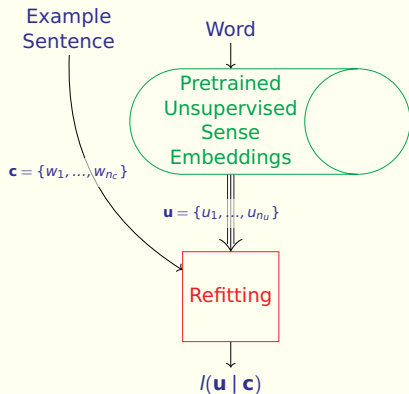
target word

# The probabilities are found using Bayes' theorem

Language model:  $P(w_i | u_i)$

Conditional Independence:

$$P(\mathbf{c} | u_i) = \prod_{\forall w_j \in \mathbf{c}} P(w_j | u_i)$$



# The probabilities are found using Bayes' theorem

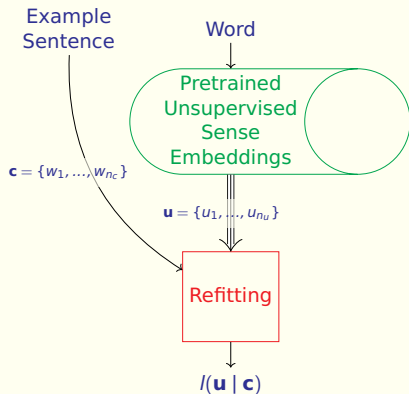
Language model:  $P(w_i | u_i)$

Conditional Independence:

$$P(\mathbf{c} | u_i) = \prod_{\forall w_j \in \mathbf{c}} P(w_j | u_i)$$

Bayes Theorem:

$$P(u_i | \mathbf{c}) = \frac{P(\mathbf{c} | u_i)P(u_i)}{\sum_{u_j \in \mathbf{s}} P(\mathbf{c} | u_j)P(u_j)}$$



$\mathbf{c}$

$w_1$   $w_2$   $w_3$   $w_4$   $w_5$   $w_6$   $w_7$   $w_8$   $w_9$   $w_{10}$

wow the wool from the kid is so soft and fluffy

target word

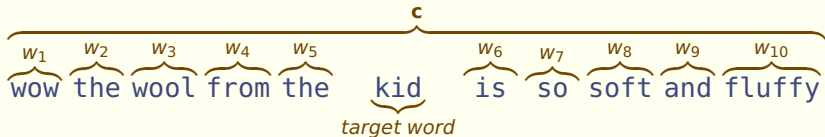
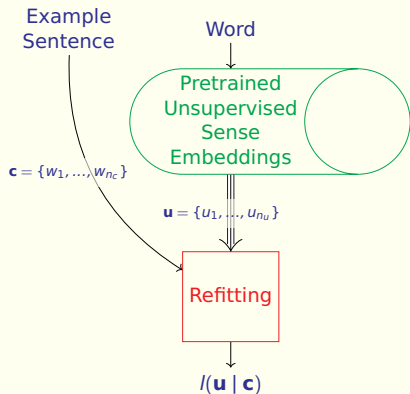
# The probabilities are found using Bayes' theorem

Bayes Theorem:

$$P(u_i | \mathbf{c}) = \frac{P(\mathbf{c} | u_i)P(u_i)}{\sum_{u_j \in \mathbf{s}} P(\mathbf{c} | u_j)P(u_j)}$$

Refitted Sense Embedding:

$$I(\mathbf{u} | \mathbf{c}) = \sum_{\forall u_i \in \mathbf{u}} u_i P(u_i | \mathbf{c})$$



The posterior distribution (over senses) is too sharp, so we smooth it

Original:

Context Likelihood:

$$P(\mathbf{c} \mid u_i) = \prod_{\forall w_j \in \mathbf{c}} P(w_j \mid u_i)$$

Sense Likelihood:

$$P(u_i \mid \mathbf{c}) = \frac{P(\mathbf{c} \mid u_i)P(u_i)}{\sum_{u_j \in \mathbf{u}} P(\mathbf{c} \mid u_j)P(u_j)}$$

The posterior distribution (over senses) is too sharp, so we smooth it

Original:

Context Likelihood:

$$P(\mathbf{c} \mid u_i) = \prod_{\forall w_j \in \mathbf{c}} P(w_j \mid u_i)$$

Sense Likelihood:

$$P(u_i \mid \mathbf{c}) = \frac{P(\mathbf{c} \mid u_i)P(u_i)}{\sum_{u_j \in \mathbf{u}} P(\mathbf{c} \mid u_j)P(u_j)}$$

Smoothed:

Context Likelihood:

$$P_S(\mathbf{c} \mid u_i) = \prod_{\forall w_j \in \mathbf{c}} \sqrt[|\mathbf{c}|]{P(w_j \mid u_i)}$$



The posterior distribution (over senses) is too sharp, so we smooth it

Original:

Context Likelihood:

$$P(\mathbf{c} | u_i) = \prod_{\forall w_j \in \mathbf{c}} P(w_j | u_i)$$

Sense Likelihood:

$$P(u_i | \mathbf{c}) = \frac{P(\mathbf{c} | u_i)P(u_i)}{\sum_{u_j \in \mathbf{u}} P(\mathbf{c} | u_j)P(u_j)}$$

Smoothed:

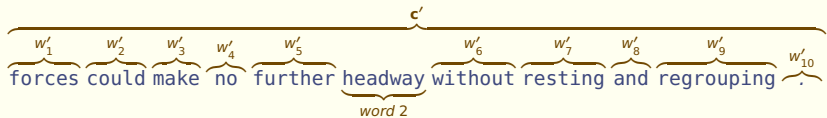
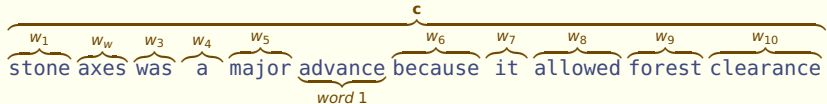
Context Likelihood:

$$P_S(\mathbf{c} | u_i) = \prod_{\forall w_j \in \mathbf{c}} \sqrt[|\mathbf{c}|]{P(w_j | u_i)}$$

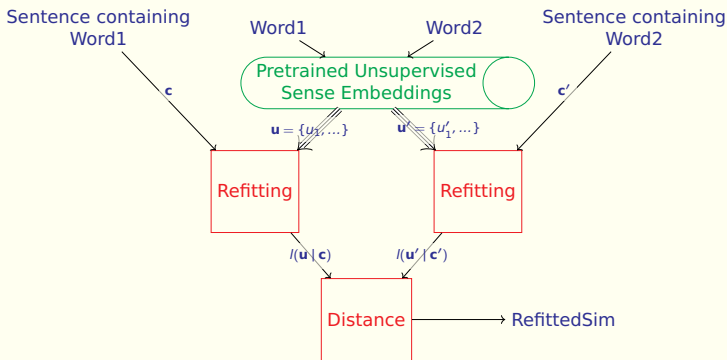
Sense Likelihood:

$$P_S(s_i | \mathbf{c}) = \frac{\sqrt[|\mathbf{c}|]{P(\mathbf{c} | u_i)P(u_i)}}{\sum_{u_j \in \mathbf{u}} \sqrt[|\mathbf{c}|]{P(\mathbf{c} | u_j)P(u_j)}}$$

# Similarity with context, is the task of ranking how similar a word is, given its usage



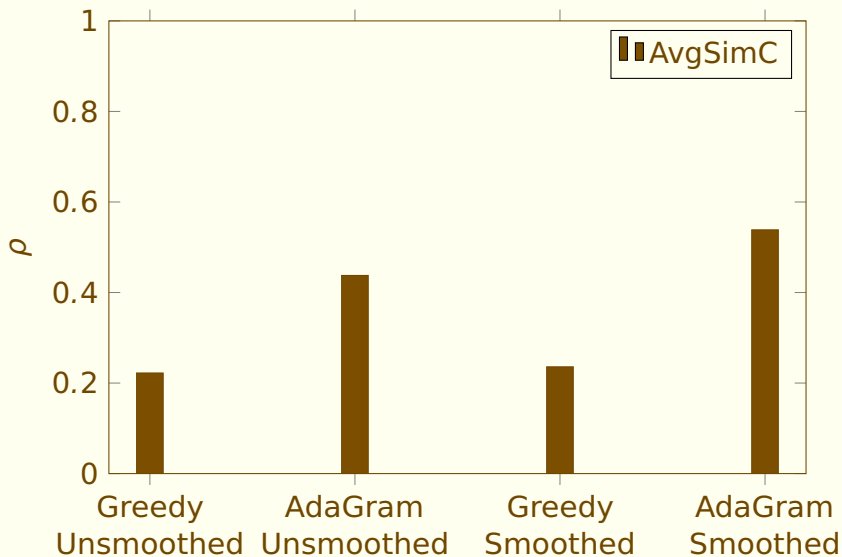
# Use for word similarity with context



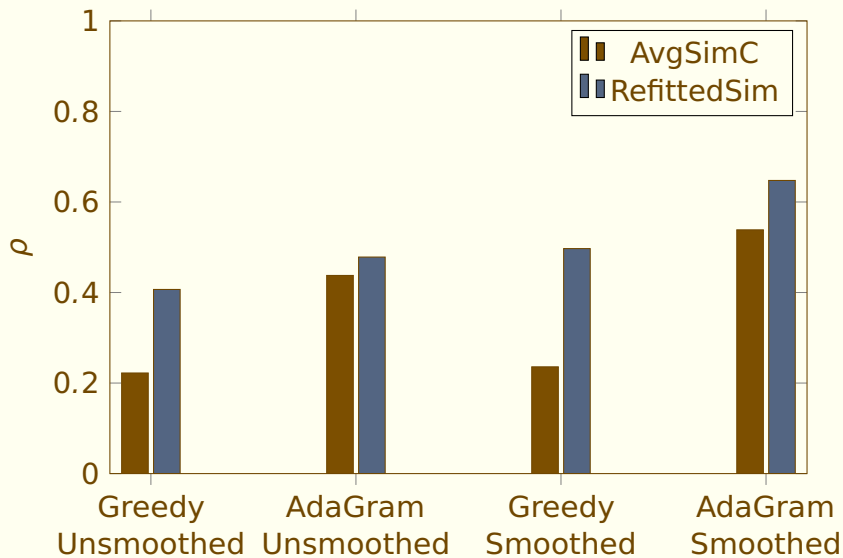
$$\text{RefittedSim}((\mathbf{u}, \mathbf{c}), (\mathbf{u}', \mathbf{c}')) = d(l(\mathbf{u} | \mathbf{c}), l(\mathbf{u}' | \mathbf{c}'))$$

$$\text{RefittedSim}((\mathbf{u}, \mathbf{c}), (\mathbf{u}', \mathbf{c}')) = d\left(\sum_{u_i \in \mathbf{u}} u_i P(u_i | \mathbf{c}), \sum_{u'_j \in \mathbf{u}'} u_j P(u'_j | \mathbf{c}'))\right)$$

## Results on word similarity with context



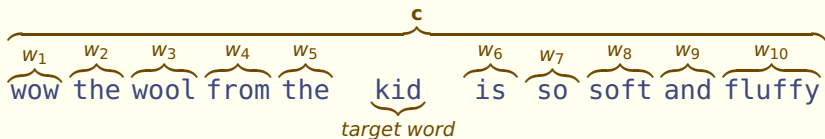
## Results on word similarity with context



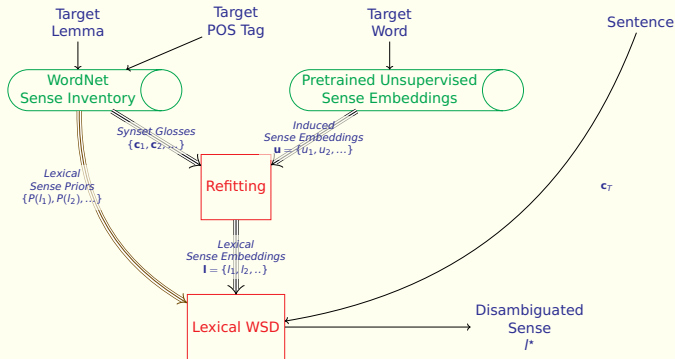
# WSD is the task of determining which sense is being used

## Kid (*Noun*)

1. (a young person of either sex) "she writes books for children"; "they're just kids"; "'tiddler' is a British term for youngster"
2. (English dramatist (1558-1594))
3. (a human offspring (son or daughter) of any age) "they had three children"; "they were able to send their kids to college"
4. (young goat)



# Use of refitted senses for word sense disambiguation

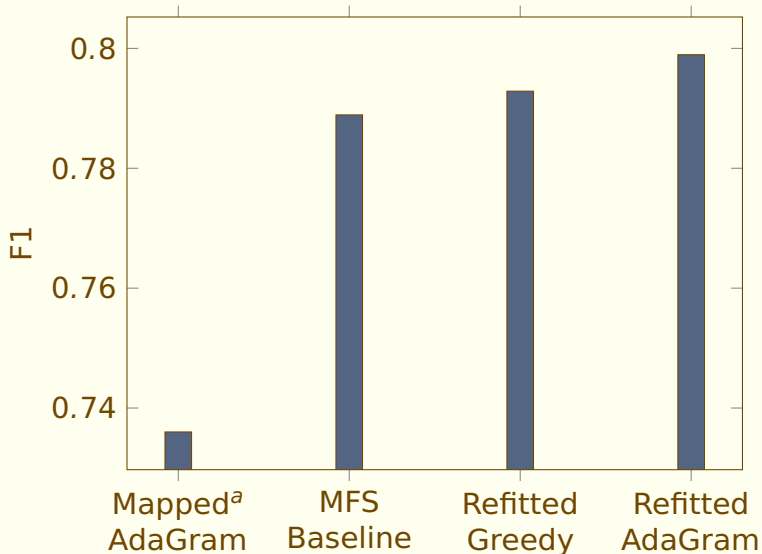


$$l^*(\mathbf{l}, \mathbf{c}_T) = \arg \max_{\forall l_i \in \mathbf{l}} P(l_i | \mathbf{c}_T)$$

$$l^*(\mathbf{l}, \mathbf{c}_T) = \arg \max_{\forall l_i \in \mathbf{l}} \frac{P(\mathbf{c}_T | l_i) P(l_i)}{\sum_{\forall l_j \in \mathbf{l}} P(\mathbf{c}_T | l_j) P(l_j)}$$

# Results for word sense disambiguation

## SemEval 2007 Task 7

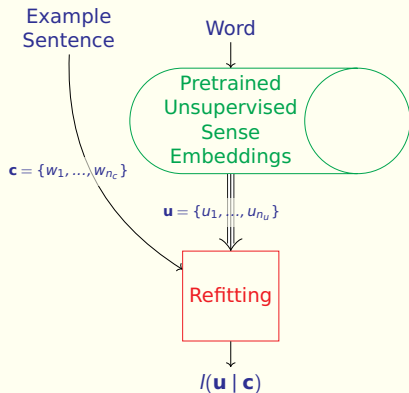


<sup>a</sup>Agirre et al. 2006



We will solve this by *refitting* them to be for the sense we mean

- ▶ Refitting constructs new sense embeddings out of the old.
- ▶ It uses the probabilities of example sentence occurring.
- ▶ The new embedding aligns to the meaning in that sentence.



# Appendix

# RefittedSim vs AvgSimC

## RefittedSim

$$\text{RefittedSim}((\mathbf{u}, \mathbf{c}), (\mathbf{u}', \mathbf{c}')) = d \left( \sum_{u_i \in \mathbf{u}} u_i P(u_i | \mathbf{c}), \sum_{u'_j \in \mathbf{u}'} u_j P(u'_j | \mathbf{c}') \right)$$

Time Complexity:  $O(n \|\mathbf{c}\| + n' \|\mathbf{c}'\|)$

## AvgSimC

$$\text{AvgSimC}((\mathbf{u}, \mathbf{c}), (\mathbf{u}', \mathbf{c}')) = \frac{1}{n \times n'} \sum_{u_i \in \mathbf{u}} \sum_{u'_j \in \mathbf{u}'} P(u_i | \mathbf{c}) P(u'_j | \mathbf{c}') d(u_i, u'_j)$$

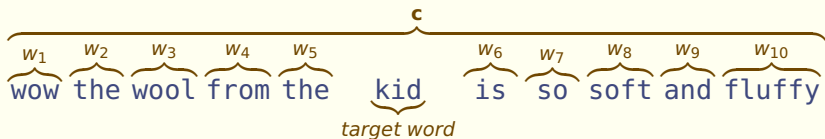
Time Complexity:  $O(n \|\mathbf{c}\| + n' \|\mathbf{c}'\| + n \times n')$

## Results on word similarity with context

Method	Geometric Smoothing	Use Prior	AvgSimC	RefittedSim
AdaGram	T	T	<b>53.8</b>	64.8
AdaGram	T	F	36.1	<b>65.0</b>
AdaGram	F	T	43.8	47.8
AdaGram	F	F	20.7	24.1
Greedy	T	F	23.6	49.7
Greedy	F	F	22.2	40.7

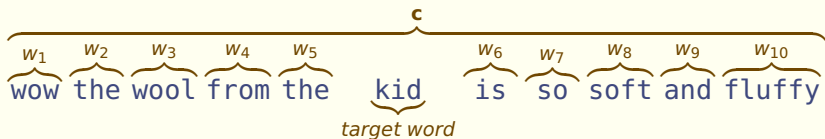
# Refitting sense-embeddings allows us to know the sense

- ▶ New embeddings are defined as a as a weighted sum of unsupervised embeddings.
- ▶ The weights are determined using the language model, with a example sentence.
- ▶ This lets us find embedding for the sense of the word in that sentence.



# Refitting sense-embeddings allows us to know the sense

- ▶ New embeddings are defined as a as a **weighted sum** of unsupervised embeddings.
- ▶ The **weights** are determined using the **language model**, with a **example sentence**.
- ▶ This lets us find embedding for the sense of the word in **that sentence**.
- ▶ Applications for **similarity with context**, and lexical tasks, such as **Word Sense Disambiguation**.



# References



E. Agirre et al. “Evaluating and optimizing the parameters of an unsupervised graph-based WSD algorithm”. In: *Proceedings of the first workshop on graph based methods for natural language processing*. Association for Computational Linguistics. 2006, pp. 89–96.



S. Bartunov et al. “Breaking Sticks and Ambiguities with Adaptive Skip-gram”. In: *CoRR* abs/1502.07257 (2015).



E. H. Huang et al. “Improving word representations via global context and multiple word prototypes”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. Association for Computational Linguistics. 2012, pp. 873–882.



G. A. Miller. “WordNet: a lexical database for English”. In: *Communications of the ACM* 38.11 (1995), pp. 39–41.