Dependency-Based Word Embeddings

By Omer Levy and Yoav Goldberg

Presented by Kaleigh Douglas

Overview

Key Motivations

- “Seek a representation that captures semantic and syntactic similarities between words.”
- Distributional Hypothesis: Words in similar contexts have similar meanings.

Key Research Questions

- How best to define context?
- Can syntactic contexts produce more focused embeddings?

Overview

Key Contributions

- Generalize Skip-Gram model from linear contexts to “arbitrary” contexts
- Experiment with syntactic contexts derived from dependency parse-trees
- Demonstrate introspection method for exploring learned contexts
- Conclude that dependency-based embeddings are less topical and more functional than the original linear-based embeddings

Methods: Skip-Gram Model with Negative Sampling

Objective: To predict probability that (word, context) pair occurs in the data

Training: Increase similarity between positive pairs and decrease similarity between negative pairs

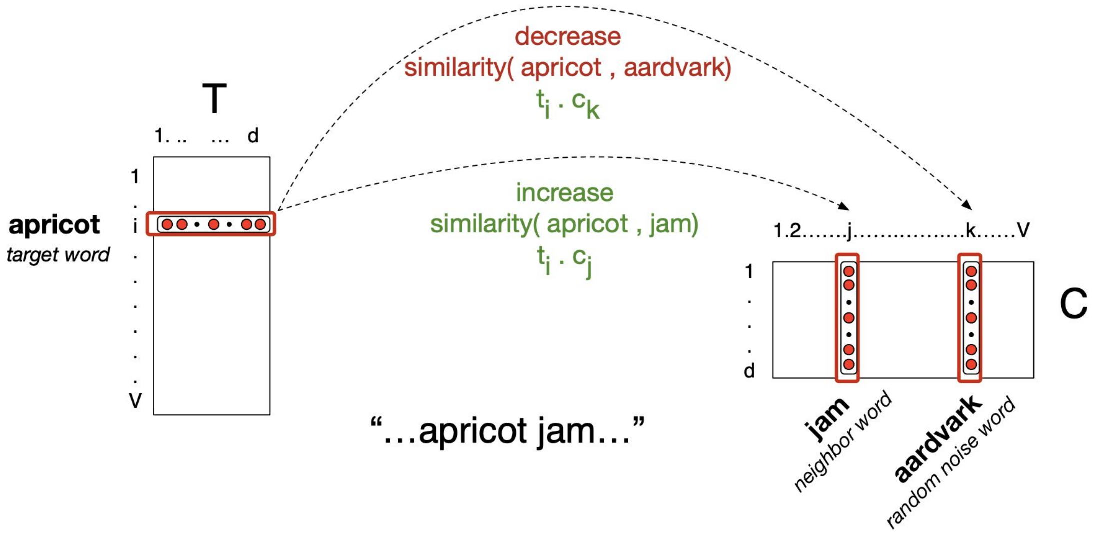


Image Credit: Jurafsky and Martin

Linear Contexts

- Defined as the “words that precede and follow the target word, typically in a window of k tokens to each side”
- Bag-of-Words
- Larger k ⇨ more topical embeddings
- Smaller k ⇨ more functional embeddings

Example (BoW k=2)

“Australian scientist **discovers** star with telescope”

[Australian, scientist, star, with]

Dependency-based Syntactic Contexts

- Created from dependency-based parse-trees
- Not limited by distance from target word - able to identify dependencies between words that are far from each other in text
- Filter out nearby words that are not directly related to target word

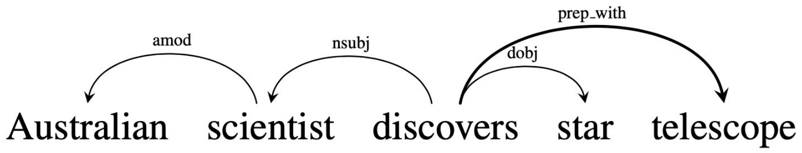
Example

“Australian scientist **discovers** star with telescope”

[scientist/nsubj, star/dobj, telescope/prep_with]

If target word = **star**

[discovers/dobj^{-1}]



Experimental Setup

- Dataset: English Wikipedia
- Model: Word2Vec (modified)
- Contexts: BoW (k=5), BoW (k=2), Dependency-based
- Negative sampling parameter: 15 negative contexts per positive context
- Embedding dimension size: 300
- Preprocessing: lowercase tokens, filtered infrequent words and tokens (< 100)

Dependency-based contexts:

- Parsed using Stanford tagger
- 175,000 words and 900,000 contexts

Qualitative Results

- Relatedness: Topical Similarity
- Similarity: Functional Similarity
- BoW contexts produce more related pairs (particularly with larger contexts)
- Dependency-based contexts produce more similar pairs

Target Word	BoW (k=5)	Dependency
batman	nightwing aquaman catwoman superman	superman superboy supergirl catwoman
hogwarts	dumbledore hallows half-blood malfoy	sunnydale collinwood calarts greendale
turing	nondeterministic non-deterministic computability deterministic	pauling hotelling heting lessing

Most similar words by cosine similarity

Quantitative Results

Task: Rank ‘similar’ pairs above ‘related’ pairs

- WordSim353 Dataset (a)
- Chiarello et al. Dataset (b)
- Pairs ranked using cosine similarity of embeddings
- Results are reversed when ranking ‘related’ pairs above ‘similar’ pairs

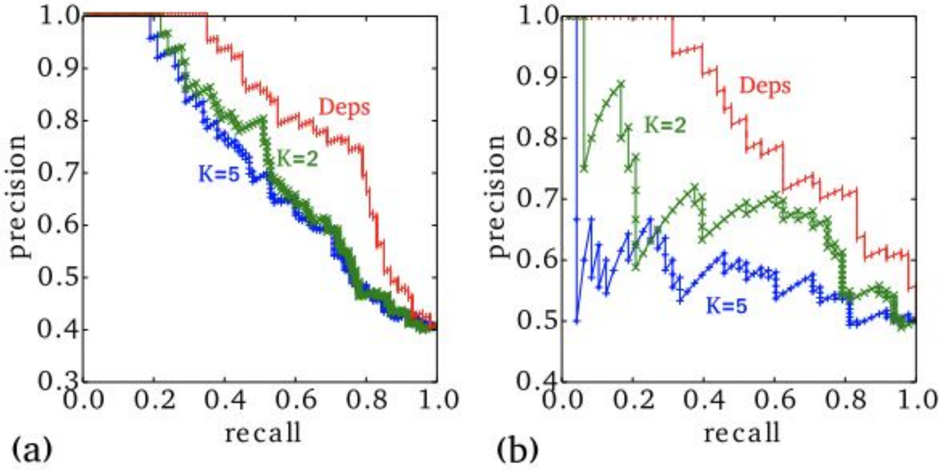


Chart Credit: Omer Levy and Yoav Goldberg

Model Introspection

- Model tries to maximize $v_c \cdot v_w$ for positive (word, context) pairs and minimize it for negative pairs
- Keep context embeddings to compute $v_c \cdot v_w$ for a specific target word with the context matrix
- Highest values indicate "most activated" contexts, which are the most discriminative contexts for target word

Target Word	Top Deps Contexts
batman (superheroes)	superman/conj^{-1} spider-man/conj^{-1} superman/conj spider-man/conj
hogwarts (schools)	students/prep_at^{-1} educated/prep_at^{-1} student/prep_at^{-1} stay/prep_at^{-1}
turing (scientists)	machine/nn^{-1} test/nn^{-1} theorem/poss^{-1} machines/nn^{-1}

Top syntactic contexts for target word (with category of top similar words from slide 8)

Ideas for Future Research

- Skip-Gram with different variations of contexts
- Determine why different grammatical relations appeared to be more important than others (compare introspection results with different categories of target words)
- Filter context for certain grammatical relation types and compare results
- Experiment with weighting scheme for contexts with different grammatical relation types
- Perform model introspection with a more structured approach (as to which words are tested)

Opinion

Overall:

- A detailed and well explained paper
- Logical extension of the Skip-Gram model

Critiques:

- Does not specify exact dataset used
- Use ‘dot product’ and ‘cosine similarity’ terms interchangeably
- Minimal testing of model introspection method
- Provided little intuition regarding choice of dependency-tree parser
- Lacked examples of tasks where functional similarity is preferred

Retrofitting Word Vectors to Semantic Lexicons

Advanced Topics in Computational Semantics

Oliviero Nardi

April 2, 2020

Introduction

- We will talk about the paper *Retrofitting Word Vectors to Semantic Lexicons* (Faruqui et al, 2015)
- In this paper, the authors provide a fast, embedding-agnostic and high performing way of incorporating semantic lexicon information into word embeddings
- Code for this paper is available at `https://github.com/mfaruqui/retrofitting`

Outline

- Motivation and Contributions
- Method
- Experiments
- Conclusion

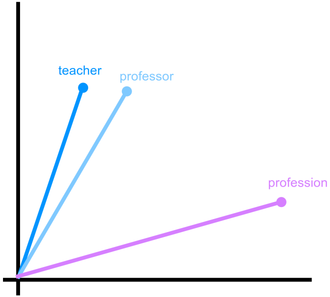
Outline

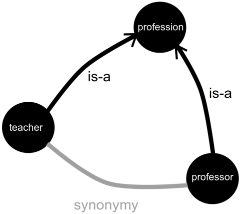
- **Motivation and Contributions**
- Method
- Experiments
- Conclusion

Motivation (1)

- **Distributional word embeddings** are among the most popular approaches in computational semantics
  - Data-driven, statistical-based vectors
  - *"Similar words appear in similar contexts"*
  - `Word2Vec`, `GloVe`...
- Another way to represent meaning is through **semantic lexicons**:
  - Network of semantic relations
  - *Hyponymy, antonymy, synonymy...*
  - `WordNet`, `FrameNet`...

Motivation (2)





How can we combine these two approaches?

Key Contributions

- Method to incorporate **semantic lexicon** information into **word embeddings**
  - Encourage **linked words** to have **similar vector representation**
- This method is **independent** of how the input vectors were constructed
- Fast **post-processing** of pre-trained word embeddings
- On a set of standard evaluation tasks:
  - **Improves** the input word embeddings
  - **Outperforms** prior incorporation approaches

Outline

- Motivation and Contributions
- **Method**
- Experiments
- Conclusion

Retrofitting: Main Idea

- We want to refine **pre-trained word embeddings** to incorporate **semantic relations**
- Assumptions:
  - They should be similar to the original word embeddings
  - Linked words should have a similar representation
- Post-processing word vectors: **retrofitting**

Problem statement

Note: $V = \{w_1, ..., w_n\}$ *is the vocabulary.*

- **Input**:
  - A matrix $\hat{Q} = (\hat{q}_1, ..., \hat{q}_n)$ of word embeddings
  - A graph $\Omega = (V, E)$ representing semantic relations between words
- **Output**:
  - A new matrix $Q = (q_1, ..., q_n)$ of word embeddings
  - Q is obtained by updating $\hat{Q}$ with the information contained in Ω.

Notice we assumed nothing about how $\hat{Q}$ was constructed.

Understanding retrofitting (1)

- *Recall*: we want Q to be close to $\hat{Q}$ (1) and to respect semantic relations (2):
  1. q_i should be similar to $\hat{q}_i$
  2. q_i should be similar to the embeddings of adjacent words q_j
- Similarity is captured by **euclidean distance**

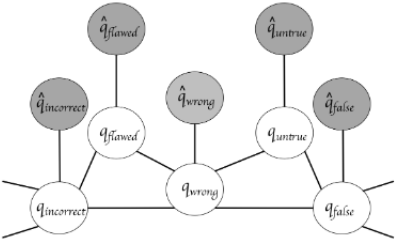


Figure 1: Word graph with edges between related words showing the observed (grey) and the inferred (white) word vector representations.

Understanding retrofitting (2)

- This yields the optimization objective

$$\mathbf{\Psi}(Q) = \sum_{i=1}^{n} [\alpha_i ||q_i - \hat{q}_i||^2 + \sum_{(i,j)\in E} \beta_{ij} ||q_i - q_j||^2] \qquad (1)$$

  where α and β tune the relative strength of associations.
- By optimizing $\mathbf{\Psi}(Q)$ we update our word embeddings
- $\mathbf{\Psi}$ is **convex** and the solution can be found by solving a **system of linear equations**:

$$q_i = \frac{\sum_{j:(i,j)\in E} \beta_{ij} q_j + \alpha_i \hat{q}_i}{\sum_{j:(i,j)\in E} \beta_{ij} + \alpha_i} \qquad (2)$$

- This can be done **quickly** in an iterative way (5 seconds for a graph of 100000 words and vector size 300)

Retrofitting vs Older approaches

- In prior approaches, the learning objective of the **word embeddings** is altered
- A **prior distribution** is added to encourage linked words to have similar vectors:

$$p(Q) \propto \exp\left(-\gamma \sum_{i=1}^{n} \sum_{j:(i,j)\in E} \beta_{ij} ||q_i - q_j||^2\right) \tag{3}$$

- Can be seen as a regularization of the embeddings (MAP estimation)
- However, this method is **not independent** of the input vectors and has no **closed form solution**

Outline

- Motivation and Contributions
- Method
- **Experiments**
- Conclusion

Experimental setup: Input data

- **Word Embeddings:**
  - `GloVe` (GLOVE)
  - `SkipGram` (SG)
  - `Global Context Vectors` (GC)
  - `Multilingual Vectors` (MULTI)
- **Semantic Lexicons:**
  - `Paraphrase DataBase` (PPDB)
  - `WordNet` (WN_{syn} and WN_{all})
  - `FrameNet` (FN)

Experimental setup: Tasks

- **Word Similarity**: Cosine similarity against human-annotated corpora: MEN-3K, RG-65 and WS-353.
- **Syntactic Relations** (SYN-REL): Find *d* such that "*a* is to *b* what *c* is to *d*". For example: "*king* is to *queen* what *actor* is to?"
- **Synonym Selection** (TOEFL): Given a target word t and four possible synonyms $\{s_1, s_2, s_3, s_4\}$, find the closest s_i to t.
- **Sentiment Analysis** (SA): Train a binary classifier on movie reviews using Q.

Experimental setup: Experiments

- Each of the **word embeddings** was retrofitted using each of the **semantic lexicons**.
- The authors tested:
  - The improvement gained by doing retrofitting
  - Performance against prior methods
  - Generalization over new languages

Results: Improvements of Retrofitting

Lexicon	MEN-3k	RG-65	WS-353	TOEFL	SYN-REL	SA
Glove	73.7	76.7	60.5	89.7	67.0	79.6
+PPDB	1.4	2.9	–1.2	**5.1**	–0.4	**1.6**
+WN$_{syn}$	0.0	2.7	0.5	**5.1**	–12.4	0.7
+WN$_{all}$	**2.2**	**7.5**	**0.7**	2.6	–8.4	0.5
+FN	–3.6	–1.0	–5.3	2.6	–7.0	0.0
SG	67.8	72.8	65.6	85.3	73.9	81.2
+PPDB	**5.4**	3.5	**4.4**	**10.7**	–2.3	**0.9**
+WN$_{syn}$	0.7	3.9	0.0	9.3	–13.6	0.7
+WN$_{all}$	2.5	**5.0**	1.9	9.3	–10.7	–0.3
+FN	–3.2	2.6	–4.9	1.3	–7.3	0.5
GC	31.3	62.8	62.3	60.8	10.9	67.8
+PPDB	**7.0**	6.1	2.0	**13.1**	**5.3**	**1.1**
+WN$_{syn}$	3.6	6.4	0.6	7.3	–1.7	0.0
+WN$_{all}$	6.7	**10.2**	**2.3**	4.4	–0.6	0.2
+FN	1.8	4.0	0.0	4.4	–0.6	0.2
Multi	75.8	75.5	68.1	84.0	45.5	81.0
+PPDB	**3.8**	4.0	**6.0**	**12.0**	**4.3**	0.6
+WN$_{syn}$	1.2	0.2	2.2	6.6	–12.3	**1.4**
+WN$_{all}$	2.9	**8.5**	4.3	6.6	–10.6	**1.4**
+FN	1.8	4.0	0.0	4.4	–0.6	0.2

Table 2: Absolute performance changes with retrofitting. Spearman's correlation (3 left columns) and accuracy (3 right columns) on different tasks. Higher scores are always better. Bold indicates greatest improvement for a vector type.

Results: Performance against prior methods (1)

Method	k, γ	MEN-3k	RG-65	WS-353	TOEFL	SYN-REL	SA
LBL (Baseline)	$k = \infty, \gamma = 0$	58.0	42.7	53.6	66.7	31.5	72.5
LBL + Lazy	$\gamma = 1$	–0.4	4.2	0.6	–0.1	0.6	1.2
	$\gamma = 0.1$	0.7	8.1	0.4	–1.4	0.7	0.8
	$\gamma = 0.01$	0.7	9.5	1.7	2.6	1.9	0.4
LBL + Periodic	$k = 100M$	3.8	18.4	3.6	12.0	4.8	1.3
	$k = 50M$	3.4	**19.5**	4.4	18.6	0.6	**1.9**
	$k = 25M$	0.5	18.1	2.7	**21.3**	–3.7	0.8
LBL + Retrofitting	–	**5.7**	15.6	**5.5**	18.6	**14.7**	0.9

Table 3: Absolute performance changes for including PPDB information while training LBL vectors. Spearman's correlation (3 left columns) and accuracy (3 right columns) on different tasks. Bold indicates greatest improvement.

Results: Generalization over new languages

Language	Task	SG	Retrofitted SG
German	RG-65	53.4	**60.3**
French	RG-65	46.7	**60.6**
Spanish	MC-30	54.0	**59.1**

Table 5: Spearman's correlation for word similarity evaluation using the using original and retrofitted SG vectors.

Findings Summary

- Retrofitting often **improves** the performance of embeddings on a variety of tasks
- It is competitive against prior methods, while also being **indepedent** of the embeddings and **fast**.
- Finally, it works well also in other languages

Outline

- Motivation and Contributions
- Method
- Experiments
- **Conclusion**

My Opinion

- Besides the pratical advantages, retrofitting is simple to understand and elegant
- Its modular approach conceptually decouples the problem of semantic lexicon integration from the training of the embeddings
- The mathematics are easy and the equations it yields are well-behaved
- **However**, as seen in $\mathrm{SYN\text{-}REL}$, performance in syntactic tasks may be significantly worse

Future Work

- **Antonymy**
  - How well does retrofitting capture antonymy?
  - How can we explicitly model antonymy?
- **Richer ways to incorporate graph information**
  - Node similarity
  - Edge types
- **Address the loss of performance in the** $\mathrm{SYN\text{-}REL}$ **task**
  - Test whether this is true for other "syntactic" tasks
  - Find ways to prevent this downgrade
  - Does this improve performance *overall*?

Questions?

Thank you for your attention!

Specialising Word Embeddings for Similarity or Relatedness

Douwe Kiela, Felix Hill, and Stephen Clark

Eui Yeon Jang

April 2, 2020

Outline

Motivation

Approach

Results

Conclusion

Outline

Motivation

Approach

Results

Conclusion

Motivation

- distributional word embeddings are **general purpose**
- "genuine" **similarity** and associative similarity (**relatedness**)
  - similarity: car-bike, chair-seat
  - relatedness: car-petrol, chair-table
- capture both quite well but perfect at neither - **mutually incompatible**
- explore **specialising embeddings** in similarity and relatedness

Outline

- Motivation
- Approach
- Results
- Conclusion

Approach

- assumption: embeddings can be **nudged** by including **additional semantic information**
  - raw text from English Wikipedia and newswire
  - synonyms from MyThes Thesaurus (similarity)
  - associated words from USF Assocation Norms (relatedness)
- three specialisation methods
  - joint learning
  - Graph-Based retrofitting
  - Skip-Gram retrofitting

Specialising Method: Joint Learning

- introduce additional semantic information to standard skip-gram objective
- **sampling condition**: include an additional context sampled uniformly

$$\frac{1}{T}\sum_{t=1}^{T}\left(J(w_t) + [w^a \sim \mathcal{U}_{A_{w_t}}]\log p(w^a|w_t)\right)$$

- **all condition**: include all additional contexts

$$\frac{1}{T}\sum_{t=1}^{T}\left(J(w_t) + \sum_{w^a \in A_{w_t}} \log p(w^a|w_t)\right)$$

Specialising Method: Graph-Based Retrofitting

- Faruqui et al. 2015
- first stage, train standard skip-gram

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c}\log p(w_{t+j}|w_t)$$

- second stage, update using semantic relation graph

$$\sum_{t=1}^{T}\left(\alpha_t||q_t-\hat{q}_t||^2+\sum_{(t,j)\in E}\beta_{tj}||q_t-q_j||^2\right)$$

Specialising Method: Skip-Gram Retrofitting

- first stage, train standard skip-gram

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c}\log p(w_{t+j}|w_t)$$

- second stage, update with additional context

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{w^a\in A_{w_t}}\log p(w^a|w_t)$$

Overview of Methods

- joint learning with sampled context
- joint learning with all contexts
- Graph-Based retrofitting
- Skip-Gram retrofitting

Evaluation

- intrinsic evaluations
  - SimLex-999 for similarity
  - MEN for relatedness
- extrinsic evaluations
  - TOEFL synonym selection task
  - document topic classification task

Outline

Motivation

Approach

Results

Conclusion

Results: Intrinsic Evaluation

Method	**SimLex-999**	**MEN**
Skip-gram	0.31	0.68
Joint-Thesaurus-Sampled	0.38	0.69
Joint-Thesaurus-All	0.44	0.60
Joint-Norms-Sampled	0.43	**0.72**
Joint-Norms-All	0.42	0.67
GB-Retrofit-Thesaurus	0.38	0.68
GB-Retrofit-Norms	0.32	0.71
SG-Retrofit-Thesaurus	**0.47**	0.69
SG-Retrofit-Norms	0.35	0.71

Table 1: Spearman ρ on a genuine similarity (SimLex-999) and relatedness (MEN) dataset (one training iteration).

Results: Extrinsic Evaluation

Method	**TOEFL**	**Doc**
Skip-gram	77.50	83.96
Joint-Thesaurus-Sampled	81.25	83.90
Joint-Thesaurus-All	80.00	83.56
Joint-Norms-Sampled	78.75	84.46
Joint-Norms-All	66.25	**84.82**
GB-Retrofit-Thesaurus	83.75	80.24
GB-Retrofit-Norms	80.00	80.58
SG-Retrofit-Thesaurus	**88.75**	84.55
SG-Retrofit-Norms	80.00	84.56

Table 2: TOEFL synonym selection and document classification accuracy (percentage of correctly answered questions/correctly classified documents).

Results: Observations

- **similarity**-specialised better on **SimLex-999** and **TOEFL task**
- **relatedness**-specialised better on **MEN** and **classification task**
- SG-retrofit matches or outperforms GB-retrofit
- generally **outperform** standard **general-purpose embeddings**
- observation of **curriculum learning** for similarity

Results: Curriculum Learning

Method	SimLex-999
Skip-gram	0.31
Fit-Thesaurus	0.26
Joint-Thesaurus-Sampled	0.38
Joint-Thesaurus-All	0.44
GB-Retrofit-Thesaurus	0.38
SG-Retrofit-Thesaurus	**0.47**

Table 3: Spearman ρ on a genuine similarity (SimLex-999) dataset (one training iteration).

Outline

Motivation

Approach

Results

Conclusion

Summary

- introduce methods of specialising embeddings
- compare different approaches on intrinsic and extrinsic tasks
- highlight ‘shortcoming’ of general-purpose embeddings
- demonstrate the advantages of specialised embeddings

My Opinion

- clear empirical evidence
- many papers specialising for different context types
- different best method for similarity and relatedness
- similarity is more difficult to learn (Hill et al., 2015)

Future Research

- fall in performance when using GB-retrofit
- different (richer) semantic information for GB/SG-retrofit
- performance of ‘universal’ representation of specialised embeddings

Questions?

Thanks for your attention!

Number of Iterations for Retrofit

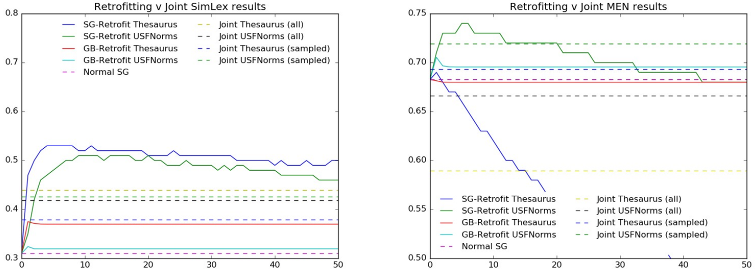


Figure 1: Varying the number of iterations when retrofitting