word2vec

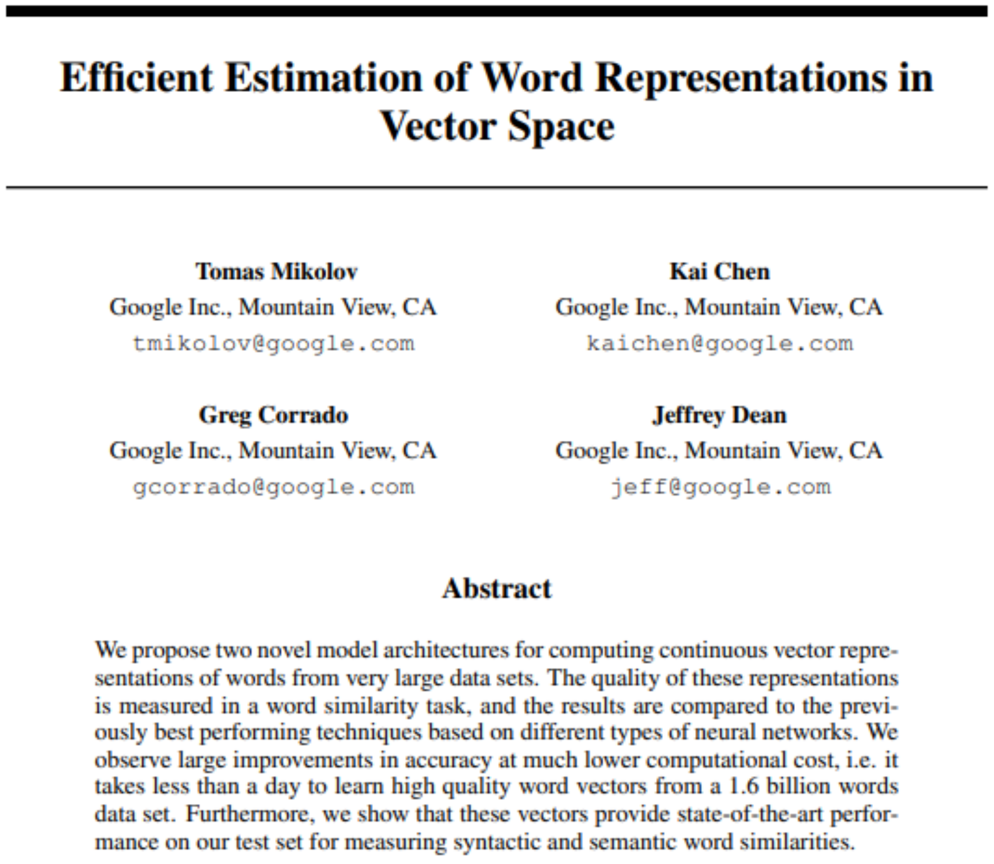
Goals

1. Appreciating the anatomy of word2vec
2. Understanding the various steps to train ad hoc embeddings
3. Assigning a role to new forms of embeddings

word2vec basics

Background

- In 2013, it drew a line between old-school and modern NLP.
- It does not require hand-labelled supervision.
- It is easy and fast to train (you can do that with Gensim).
- It is OSS.



Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

Philosophy

- It trains a classifier on a binary prediction task:
  - “is word **ω** likely to show up near word **η**”?
- The classification task is 'instrumental' in nature:
  - The point is not predicting the 'next' word.
  - The goal is to adjust word vectors.

The algorithm

Caveat: there are various word2vec flavors — here, we focus on the Skip-Gram (SG) flavor

Skip-Gram algorithm

The algorithm has four main steps:

1. Treat the target word and a neighbouring context word as positive examples.
2. Randomly sample other words in the lexicon to get negative examples.
3. Use logistic regression to train a classifier to distinguish those two cases.
4. Use the weights as the embeddings.

SG training data

Let us consider the target word ‘apricot’ and its context lexemes based on a window of length ‘2’:

not in context	**c1**	**c2**	**t**	**c3**	**c4**	**not in context**
... lemon, a	tablespoon	of	apricot	jam	a	pinch of ...

SG goals

Given a tuple **(t, c) = target, context**, we want to estimate the probability that **c** is a real context word for **t**, namely

$$P(+|t,c)$$

How to compute P(+|t, c)?

Intuition

- Words are likely to appear near similar words.
- Hence, it is reasonable to model word similarity as the dot-product of the underlying vectors **t • c**

Problem

- Dot product is not a probability!
- Neither is cosine.

Turning dot product into a probability

$$P(+|t,c) = \frac{1}{1+e^{-t \bullet c}}$$

$$P(-|t,c) = 1 - P(+|t,c) = \frac{e^{-t \bullet c}}{1+e^{-t \bullet c}}$$

For all context words

$$P(+|t,c_{1:k}) = \prod_{i=1}^{k} \frac{1}{1+e^{-t \bullet c_i}}$$

Let us see SG in action

SG training (1/2)

Given the sentence:

not in context	c1	c2	t	c3	c4	not in context
... lemon, a	tablespoon	of	apricot	jam	a	pinch of ...

We look for positive examples +:

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	preserve
apricot	...

SG training (2/2)

Given the list of positive words **c+**

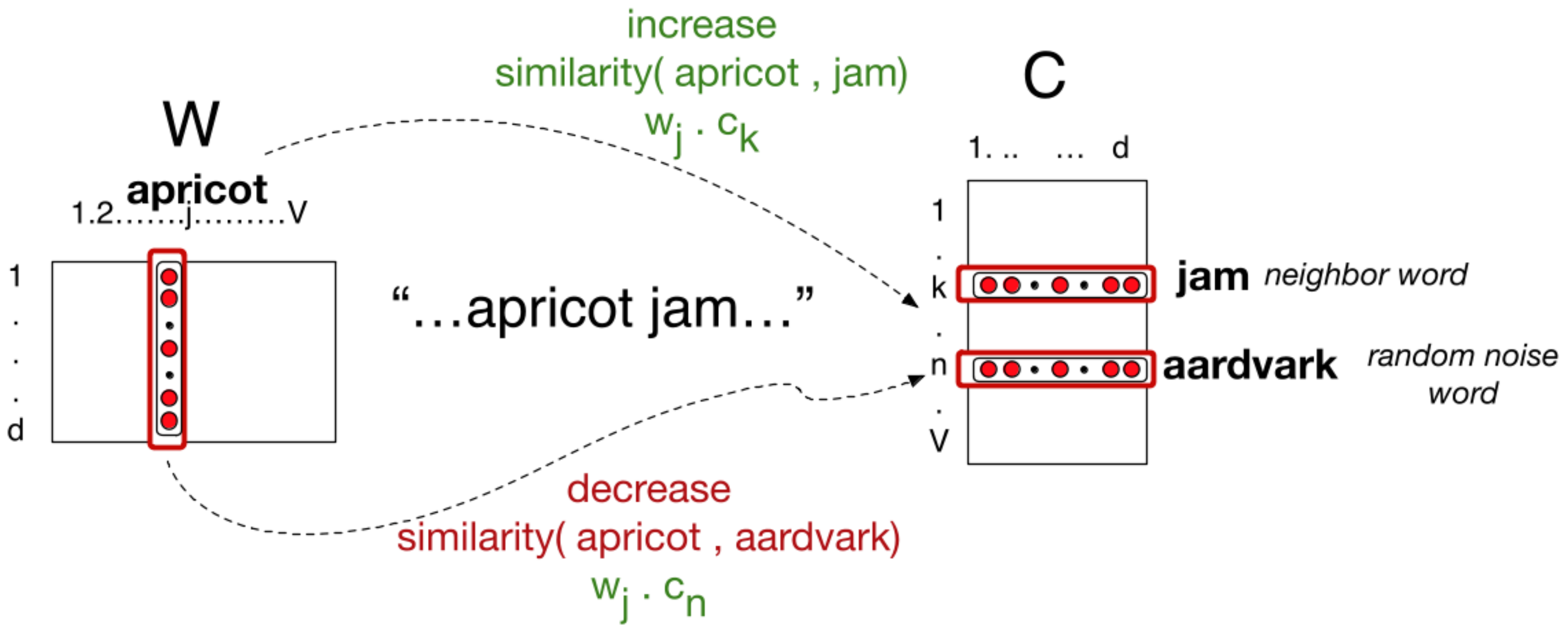
- Each positive **c** is matched with a negative **c**
- Negatives are ‘noise words’ that do not belong to any linguistic contexts of **t**

Setup

Let's represent words as vectors of some length (say 300), randomly initialised.

- We start with **300 * V** random parameters.
- Over the entire training set, we would like to adjust those word vectors such that we:
    - Maximise the similarity of the target word, context word pairs **(t, c)** drawn from the positive data.
    - Minimise the similarity of the **(t, c)** pairs drawn from the negative data.

Learning word embeddings



Source: Jurafski and Martin, 2019

Wra-up

Main points

- The word2vec algorithm draws on DH and ML to predict the likelihood that a specific context word **c** will be appear in a linguistic context conditionally on **t.**
- To do so, it regresses these conditional probabilities on the real values included in the word vectors.
- During the training, the real values are adjusted such that words that are similar in a semantic sense have similar vectors.