the morning paper

an interesting/influential/important paper from the world of CS every weekday morning, as selected by Adrian Colyer

The amazing power of word vectors

APRIL 21, 2016

tags: Google, Machine Learning

For today's post, I've drawn material not just from one paper, but from five! The subject matter is 'word2vec' – the work of Mikolov et al. at Google on efficient vector representations of words (and what you can do with them). The papers are:

- **Efficient Estimation of Word Representations in Vector Space (http://arxiv.org/pdf/1301.3781.pdf)** – Mikolov et al. 2013
- **Distributed Representations of Words and Phrases and their Compositionality (https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf)** – Mikolov et al. 2013
- **Linguistic Regularities in Continuous Space Word Representations (http://www.aclweb.org/anthology/N13-1090)** – Mikolov et al. 2013
- **word2vec Parameter Learning Explained (http://arxiv.org/pdf/1411.2738v3.pdf)** – Rong 2014
- **word2vec Explained: Deriving Mikolov et al's Negative Sampling Word-Embedding Method (http://arxiv.org/pdf/1402.3722v1.pdf)** – Goldberg and Levy 2014

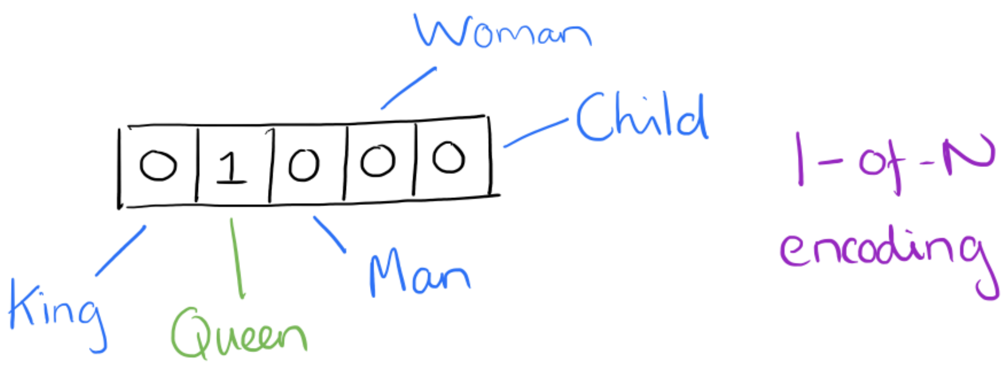
From the first of these papers ('Efficient estimation…') we get a description of the *Continuous Bag-of-Words* and *Continuous Skip-gram* models for learning word vectors (we'll talk about what a word vector is in a moment…). From the second paper we get more illustrations of the power of word vectors, some additional information on optimisations for the skip-gram model (hierarchical softmax and negative sampling), and a discussion of applying word vectors to phrases. The third paper ('Linguistic Regularities…') describes vector-oriented reasoning based on word vectors and introduces the famous "King – Man + Woman = Queen" example. The last two papers give a more detailed explanation of some of the very concisely expressed ideas in the Milokov papers.

Check out the **word2vec implementation (https://code.google.com/archive/p/word2vec)** on Google Code.

What is a word vector?

At one level, it's simply a vector of weights. In a simple 1-of-N (or 'one-hot') encoding every element in the vector is associated with a word in the vocabulary. The encoding of a given word is simply the vector in which the corresponding element is set to one, and all other elements are zero.

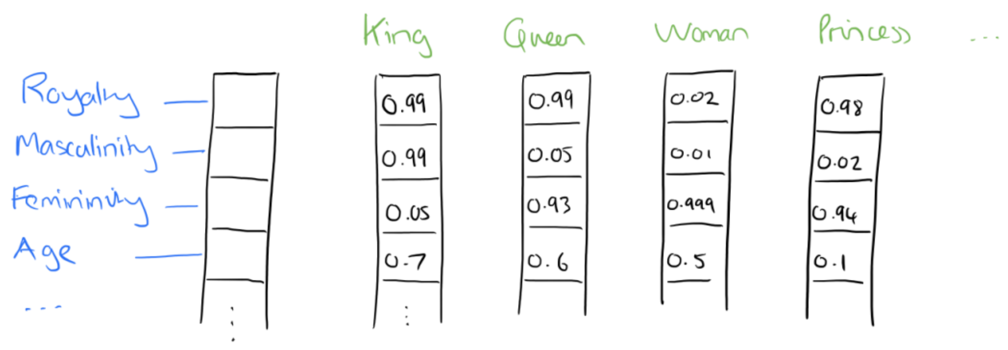
Suppose our vocabulary has only five words: King, Queen, Man, Woman, and Child. We could encode the word 'Queen' as:



Using such an encoding, there's no meaningful comparison we can make between word vectors other than equality testing.

In word2vec, a *distributed* representation of a word is used. Take a vector with several hundred dimensions (say 1000). Each word is representated by a distribution of weights across those elements. So instead of a one-to-one mapping between an element in the vector and a word, the representation of a word is spread across all of the elements in the vector, and each element in the vector contributes to the definition of many words.

If I label the dimensions in a hypothetical word vector (there are no such pre-assigned labels in the algorithm of course), it might look a bit like this:



	King	Queen	Woman	Princess	...
Royalty	0.99	0.99	0.02	0.98	
Masculinity	0.99	0.05	0.01	0.02	
Femininity	0.05	0.93	0.999	0.94	
Age	0.7	0.6	0.5	0.1	
...	⋮				

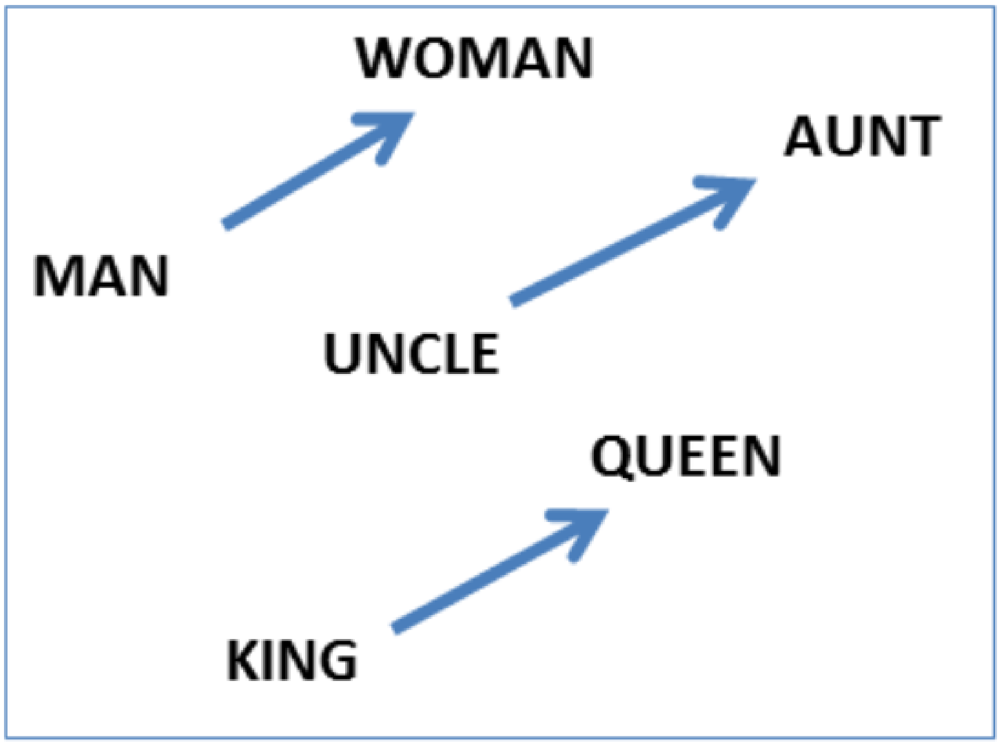
Such a vector comes to represent in some abstract way the 'meaning' of a word. And as we'll see next, simply by examining a large corpus it's possible to learn word vectors that are able to capture the relationships between words in a surprisingly expressive way. We can also use the vectors as inputs to a neural network.

Reasoning with word vectors

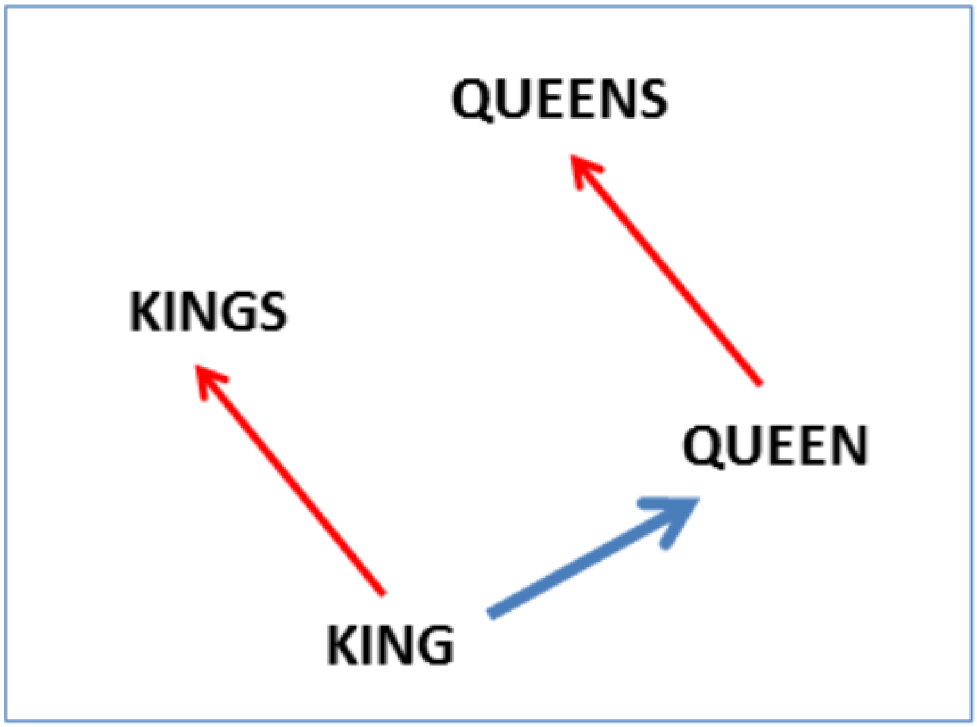
> *We find that the learned word representations in fact capture meaningful syntactic and semantic regularities in a very simple way. Specifically, the regularities are observed as constant vector offsets between pairs of words sharing a particular relationship. For example, if we denote the vector for word i as x_i, and focus on the singular/plural relation, we observe that $x_{apple} - x_{apples} \approx x_{car} - x_{cars}$, $x_{family} - x_{families} \approx x_{car} - x_{cars}$, and so on. Perhaps more surprisingly, we find that this is also the case for a variety of semantic relations, as measured by the SemEval 2012 task of measuring relation similarity.*

The vectors are very good at answering analogy questions of the form *a* is to *b* as *c* is to *?*. For example, *man* is to *woman* as *uncle* is to *?* (*aunt*) using a simple vector offset method based on cosine distance.

For example, here are vector offsets for three word pairs illustrating the gender relation:



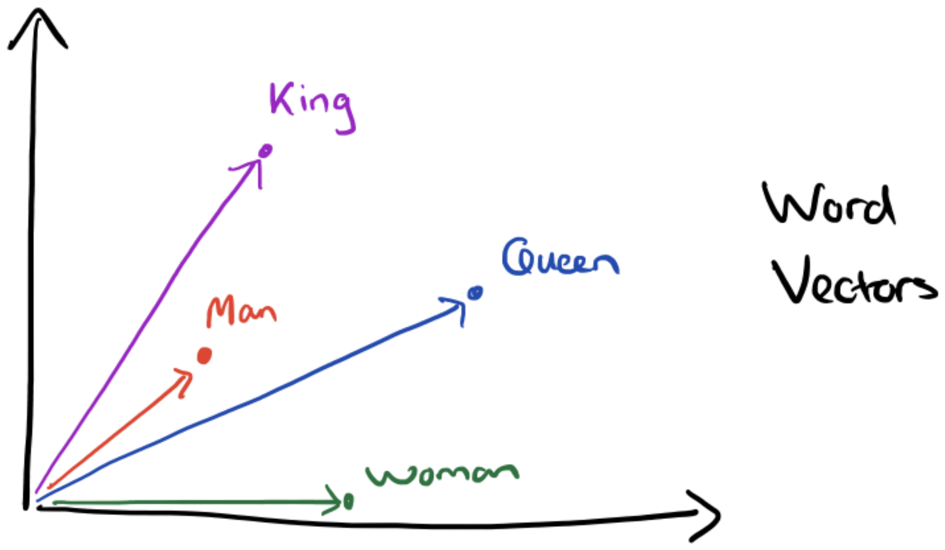
And here we see the singular plural relation:



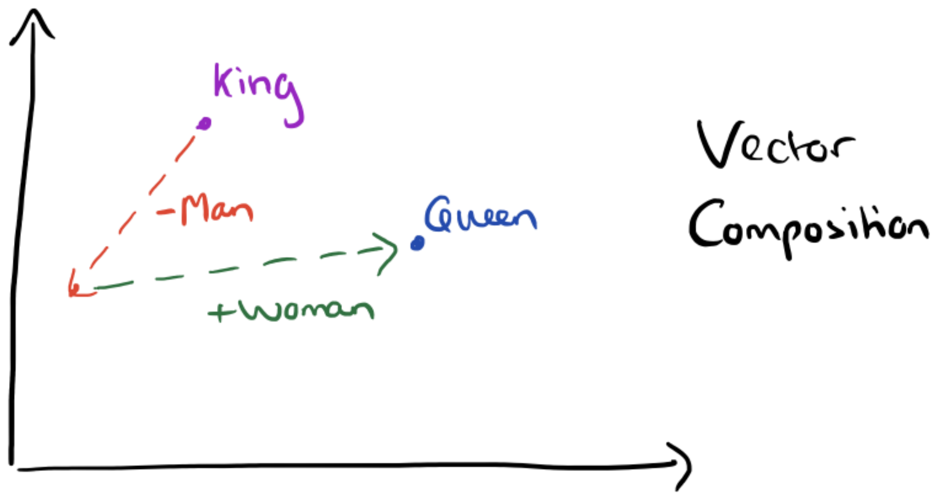
This kind of vector composition also lets us answer "King – Man + Woman = ?" question and arrive at the result "Queen" ! All of which is truly remarkable when you think that all of this knowledge simply comes from looking at lots of word in context (as we'll see soon) with no other information provided about their semantics.

> *Somewhat surprisingly, it was found that similarity of word representations goes beyond simple syntatic regularities. Using a word offset technique where simple algebraic operations are performed on the word vectors, it was shown for example that vector("King") – vector("Man") + vector("Woman") results in a vector that is closest to the vector representation of the word Queen.*

Vectors for King, Man, Queen, & Woman:

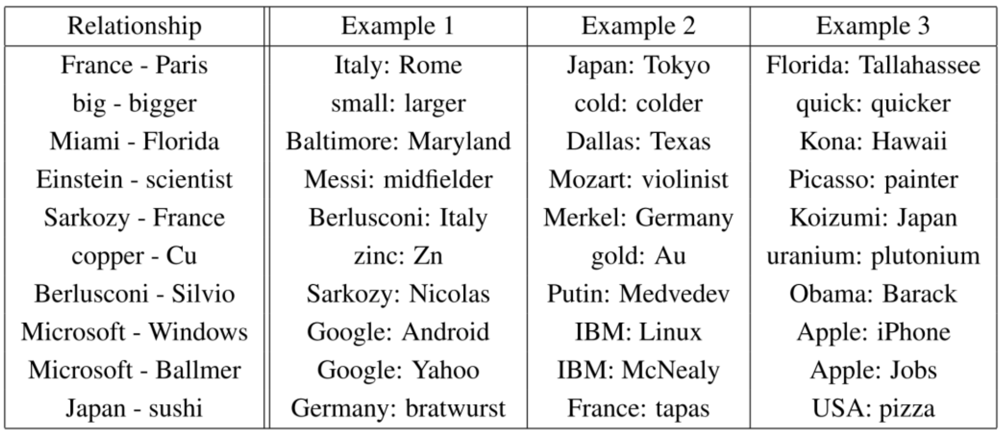


The result of the vector composition King – Man + Woman = ?



Here are some more results achieved using the same technique:

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*



Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Here's what the country-capital city relationship looks like in a 2-dimensional PCA projection:

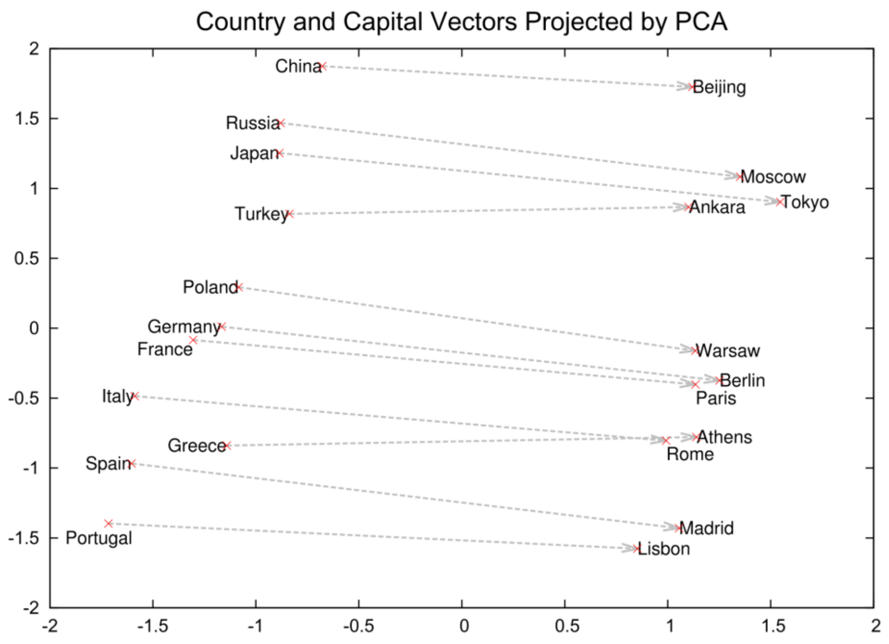
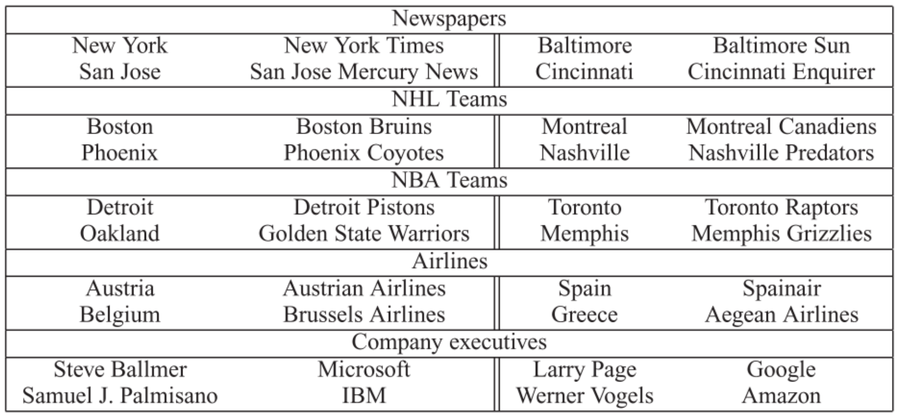


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

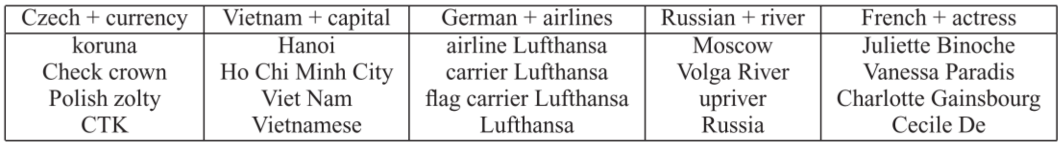
Here are some more examples of the 'a is to b as c is to ?' style questions answered by word vectors:



Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.

We can also use element-wise addition of vector elements to ask questions such as 'German + airlines' and by looking at the closest tokens to the composite vector come up with impressive answers:



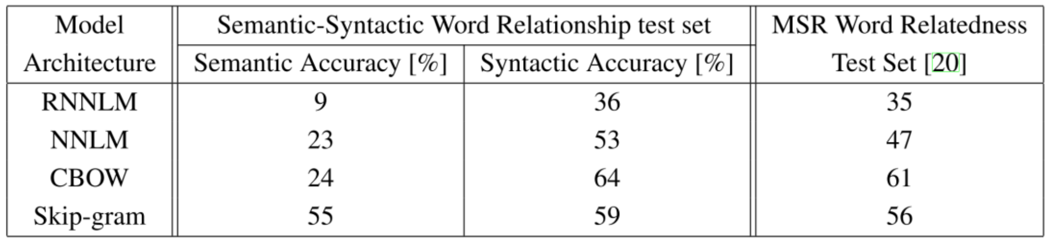
Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

> *Word vectors with such semantic relationships could be used to improve many existing NLP applications, such as machine translation, information retrieval and question answering systems, and may enable other future applications yet to be invented.*

The Semantic-Syntatic word relationship tests for understanding of a wide variety of relationships as shown below. Using 640-dimensional word vectors, a skip-gram trained model achieved 55% semantic accuracy and 59% syntatic accuracy.

Table 3: *Comparison of architectures using models trained on the same data, with 640-dimensional word vectors. The accuracies are reported on our Semantic-Syntactic Word Relationship test set, and on the syntactic relationship test set of [20]*



Model	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness
Architecture	Semantic Accuracy [%]	Syntactic Accuracy [%]	Test Set [20]
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

Learning word vectors

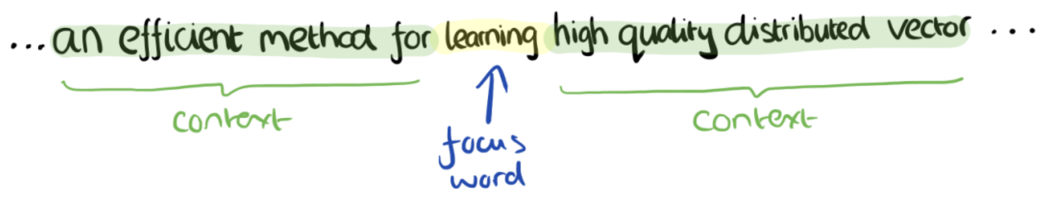
Mikolov et al. weren't the first to use continuous vector representations of words, but they did show how to reduce the computational complexity of learning such representations – making it practical to learn high dimensional word vectors on a large amount of data. For example, "We have used a Google News corpus for training the word vectors. This corpus contains about 6B tokens. We have restricted the vocabulary size to the 1 million most frequent words…"

The complexity in neural network language models (feedforward or recurrent) comes from the non-linear hidden layer(s).

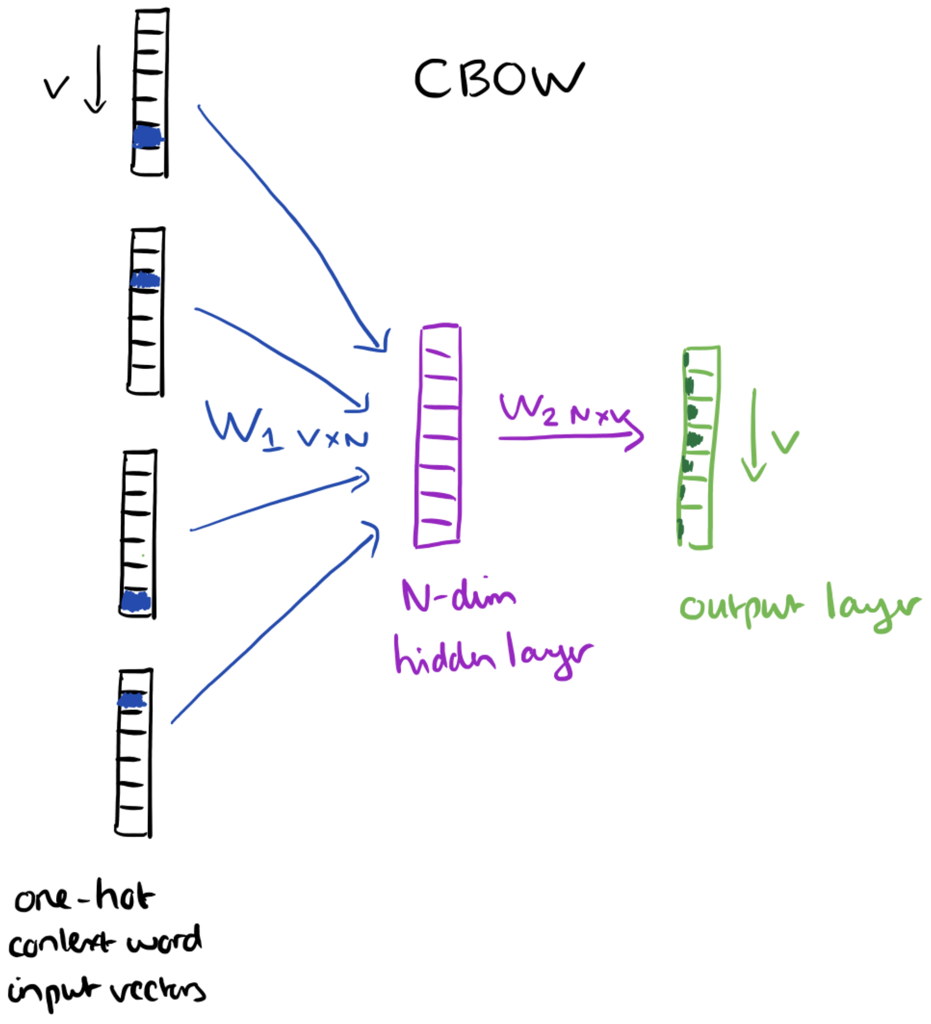
> *While this is what makes neural networks so attractive, we decided to explore simpler models that might not be able to represent the data as precisely as neural networks, but can posssible be trained on much more data efficiently.*

Two new architectures are proposed: a *Continuous Bag-of-Words* model, and a *Continuous Skip-gram* model. Let's look at the continuous bag-of-words (CBOW) model first.

Consider a piece of prose such as "The recently introduced continuous Skip-gram model is an efficient method for learning high-quality distributed vector representations that capture a large number of precises syntatic and semantic word relationships." Imagine a sliding window over the text, that includes the central word currently in focus, together with the four words and precede it, and the four words that follow it:

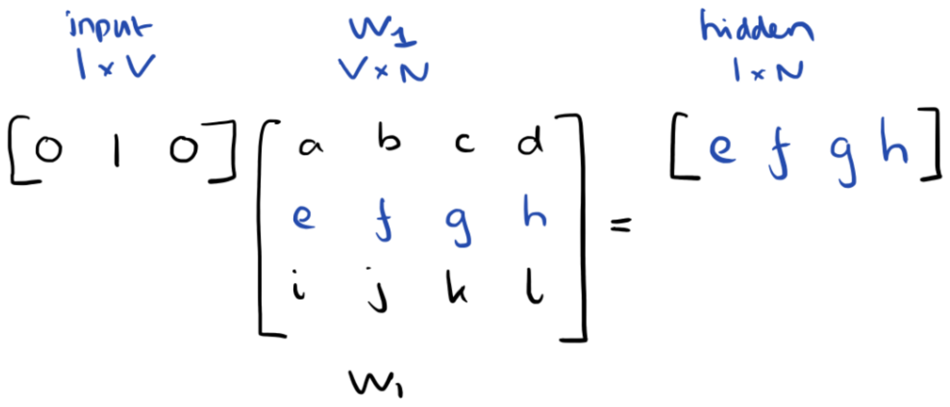

…an efficient method for learning high quality distributed vector …
context
focus word
context

The context words form the input layer. Each word is encoded in one-hot form, so if the vocabulary size is V these will be V-dimensional vectors with just one of the elements set to one, and the rest all zeros. There is a single hidden layer and an output layer.



The training objective is to maximize the conditional probability of observing the actual output word (the focus word) given the input context words, with regard to the weights. In our example, given the input ("an", "efficient", "method", "for", "high", "quality", "distributed", "vector") we want to maximize the probability of getting "learning" as the output.

Since our input vectors are one-hot, multiplying an input vector by the weight matrix $\mathbf{W_1}$ amounts to simply selecting a row from $\mathbf{W_1}$.

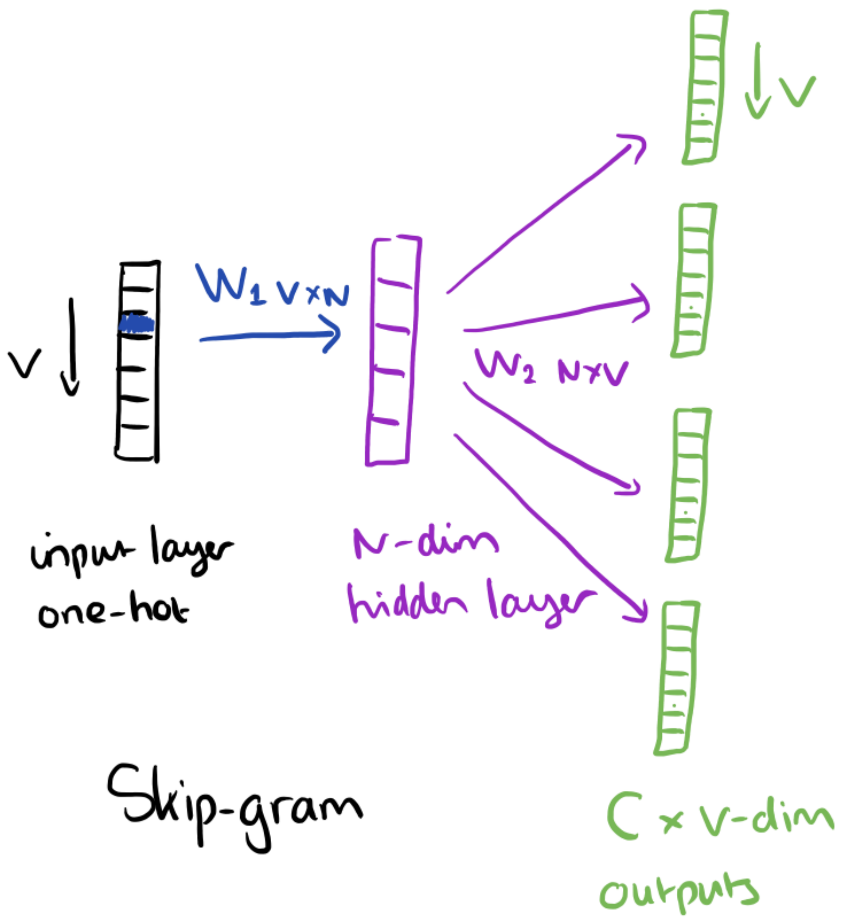


Given *C* input word vectors, the activation function for the hidden layer **h** amounts to simply summing the corresponding 'hot' rows in W_1, and dividing by C to take their average.

> *This implies that the link (activation) function of the hidden layer units is simply linear (i.e., directly passing its weighted sum of inputs to the next layer).*

From the hidden layer to the output layer, the second weight matrix $\mathbf{W_2}$ can be used to compute a score for each word in the vocabulary, and softmax can be used to obtain the posterior distribution of words.

The **skip-gram** model is the opposite of the CBOW model. It is constructed with the focus word as the single input vector, and the target context words are now at the output layer:



The activation function for the hidden layer simply amounts to copying the corresponding row from the weights matrix $\mathbf{W_1}$ (linear) as we saw before. At the output layer, we now output C multinomial distributions instead of just one. The training objective is to mimimize the summed prediction error across all context words in the output layer. In our example, the input would be "learning", and we hope to see ("an", "efficient", "method", "for", "high", "quality", "distributed", "vector") at the output layer.

Optimisations

Having to update every output word vector for every word in a training instance is very expensive….

> *To solve this problem, an intuition is to limit the number of output vectors that must be updated per training instance. One elegant approach to achieving this is hierarchical softmax; another approach is through sampling.*

Hierarchical softmax uses a binary tree to represent all words in the vocabulary. The words themselves are leaves in the tree. For each leaf, there exists a unique path from the root to the leaf, and this path is used to estimate the probability of the word represented by the leaf. "We define this probability as the probability of a random walk starting from the root ending at the leaf in question."

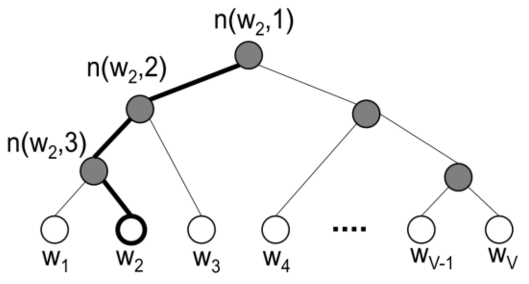


Figure 4: An example binary tree for the hierarchical softmax model. The white units are words in the vocabulary, and the dark units are inner units. An example path from root to w_2 is highlighted. In the example shown, the length of the path $L(w_2) = 4$. $n(w, j)$ means the j-th unit on the path from root to the word w.

> *The main advantage is that instead of evaluating V output nodes in the neural network to obtain the probability distribution, it is needed to evaluate only about $log_2(V)$ words… In our work we use a binary Huffman tree, as it assigns short codes to the frequent words which results in fast training.*

Negative Sampling is simply the idea that we only update a sample of output words per iteration. The target output word should be kept in the sample and gets updated, and we add to this a few (non-target) words as negative samples. "A probabilistic distribution is needed for the sampling process, and it can be arbitrarily chosen… One can determine a good distribution empirically."

Mikolov et al. also use a simple subsampling approach to counter the imbalance between rare and frequent words in the training set (for example, "in", "the", and "a" provide less information value than rare words). Each word in the training set is discarded with probability $P(w_i)$ where

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

$f(w_i)$ is the frequency of word w_i and *t* is a chosen threshold, typically around 10^{-5}.

from → Machine Learning, Uncategorized

154 Comments leave one →

1. **sp PERMALINK**
   April 21, 2016 5:10 pm
   Nice concise overview. Thanks!

   REPLY
2. **Johan Zing PERMALINK**
   April 22, 2016 8:46 am
   Nice article, and a great explanation of word2vec! I'd just like to point out that in "Linguistic Regularities in Continuous Space Word Representations", the word vectors are learned using a recursive NN (as opposed to the feed forward architecture of CBOW and Skip-Gram). One again, the vector

representations are taken from the weight matrix between input and hidden layer.

REPLY

3. **@feedly PERMALINK**
   May 31, 2016 10:28 pm
   Very nice overview. Thanks! -Edwin

   REPLY
4. **Zteve PERMALINK**
   November 4, 2016 10:53 am
   Hi Adrian,
   Just noticed a few formatting problems in the last paragraph. I'm sure you can fix 'em easily.

   REPLY
   - **adriancolyer PERMALINK***
     November 5, 2016 12:57 pm
     Fixed, thank you!

     REPLY
5. **Saurav PERMALINK**
   December 30, 2016 8:01 pm
   Probably the best explanation I found on the internet.
   Cleared all my doubts!

   REPLY
6. **rong PERMALINK**
   January 5, 2017 4:14 pm
   Hi, when using Negative Sampling,the NN's loss isn't "2-p(D=1|w,t)+p(D=0|wi,t)"? i mean the input-label as a input,not for computer loss as usual,is that correct ?

   REPLY
7. **Kirill Kovalewsky PERMALINK**
   January 9, 2017 9:20 pm
   Thank fort this overview

   REPLY
8. **Godson PERMALINK**
   January 12, 2017 11:26 am
   Awesome… Great Blog

   REPLY
9. **Irma_Ravkic PERMALINK**
   January 19, 2017 1:08 am
   Very nice, illustrative and clear.

   REPLY
10. **varun mishra PERMALINK**
    February 6, 2017 2:53 pm
    Very well explained 🙂

    REPLY
11. **Jenner Little PERMALINK**
    March 3, 2017 8:58 am

What Mikolov and Google based their work on: https://www.kaggle.com/c/word2vec-nlp-tutorial/discussion/12349

REPLY

12. **Allan Wind PERMALINK**
    March 3, 2017 5:19 pm
    The link for "Linguistic Regularities in Continuous Space Word Representations" is not found (404).

    REPLY

    - **adriancolyer PERMALINK***
      March 4, 2017 10:43 am
      Updated the link, thank you!

      REPLY

13. **David Parks PERMALINK**
    April 20, 2017 3:36 am
    Fabulous article! What did you use to produce the hand sketch drawings? They're fabulous.

    REPLY

    - **adriancolyer PERMALINK***
      April 20, 2017 7:11 am
      Thank you! The sketch drawings are done 'by hand' using an app called Notability on an iPad Pro with the Apple pencil.

      REPLY

14. **Makar PERMALINK**
    April 24, 2017 3:31 am
    Hi, Well written. Could you explain (in CBOW), from where the weight matrix (V x N) is coming from? How are we calculating this matrix?

    REPLY

15. **Xiaotong Xu PERMALINK**
    June 30, 2017 6:30 am
    Is there any info about lookup table and weight matrix?I have read a lot of papers and block about word2vec, but few about lookup table or weight matrix.Thx

    REPLY

    - **Xiaotong Xu PERMALINK**
      June 30, 2017 7:13 am
      Sorry for forgetting to describe the [lookup table] and [weight matrix]
      lookup table: W1v*n in your paper.
      weight matrix:W2n*v in your paper.

      REPLY

16. **MPQ PERMALINK**
    July 1, 2017 4:51 am
    I really want to know how we can obtain C different output matrixes of skip-gram? the hidden layer is the same(size(N * 1)), W2 is the not the same?

    REPLY

17. **Repzz PERMALINK**
    July 7, 2017 2:30 pm

Thank you for the interesting paper! Just wanted to notice that there is a mistake in the Mikolov's name in the last paragraph. Best regrds!

REPLY

- **adriancolyer PERMALINK***
  July 8, 2017 8:14 am
  Fixed, thank you!

  REPLY

  - **Nikhil Goel PERMALINK**
    August 4, 2017 4:10 pm
    Exceptional Explanation to a complex topic.. Best Article read ever on W2Vec

18. **Abhi PERMALINK**
    September 18, 2017 3:45 pm
    Thanks

    REPLY
19. **Miguel PERMALINK**
    November 8, 2017 12:14 pm
    Thanks for the intuitive description of word2vec, and the CBOW and skip-gram models! 🙂 It would have been nice if you had tied the two concepts together at the end. It is not clear from your post how the output of the CBOW/skip-gram neural network is converted into a single, lower dimensional vector representation of a given word.

    REPLY
20. **Radek Skowron PERMALINK**
    February 24, 2018 6:38 pm
    I have found this article thanks to SEO by the Sea. Frankly, I didn't know about "word vectors" and that's why I was quite interested how it's actually works. Your article sheds some light about the thing, thanks!

    REPLY
21. **MT PERMALINK**
    February 27, 2018 6:03 pm
    syntactic not syntatic 😉 Otherwise great post!

    REPLY
22. **Jon M PERMALINK**
    April 30, 2018 10:47 am
    Excellent overview. Filled in some gaps. Thanks

    REPLY
23. **Nisar Khan PERMALINK**
    October 16, 2018 5:46 pm
    Hi Adrian,
    Instead of the "gender" relation analogy example { man:woman} implying { uncle:aunt, king:queen} , it would be good if we can instead have {uncle:aunt } implying { king:queen, grandpa:grandma } and the relation as "married". Because {man: woman} "gender" analogy can also imply {uncle:grandma, king:princess} etc.,
    Best Regards.

    REPLY

Trackbacks

1. Big Analytics Roundup (April 25, 2016) | The Big Analytics Blog
2. 2016/04/25 ML Reddit – cuponthetop
3. Distributed representations of sentences and documents | the morning paper
4. Sequence to sequence learning with neural networks | the morning paper
5. Building end-to-end dialogue systems using generative hierarchical neural network models | the morning paper
6. Natural language understanding (almost) from scratch | the morning paper
7. End of Term, and the power of compound interest | the morning paper
8. Word Embedding – badripatro
9. GloVe: Global Vectors for Word Representation – quyv
10. Distributed representations of sentences and documents – quyv
11. Word2Vec – Learning the meaning behind words | One-Tech-A-Day
12. Deep neural networks for YouTube recommendations | the morning paper
13. COGNITIONX DATA SCIENCE, AI AND MACHINE LEARNING BRIEFING ISSUE 21 - CognitionX
14. word vectors | What I learned
15. Achieving human parity in conversational speech recognition | the morning paper
16. So that was 2016 | the morning paper
17. Word Vector Representation: Word2Vec & Glove
18. The Brain’s Registers | Pointers Gone Wild
19. Teaching Machines to Read Emails: Feature Selection | Stacks and Q's
20. &quot;King – Man + Woman = ?&quot; | CS130 Journals
21. “King – Man + Woman = ?” | CS130 Journals
22. Understanding, generalisation, and transfer learning in deep neural networks | the morning paper
23. Unsupervised learning and GANs | the morning paper
24. The amazing power of word vectors – thoughts…
25. Word Vectors (word2vec) | Ace Infoway
26. GloVe: Global Vectors for Word Representation | the morning paper
27. Cooking receipts (I) | The Data Explorer
28. Word (Thought) Vectors | Delightful & Distinctive COLRS
29. pupper2vec: analysing internet dog slang with machine learning | gutterstats
30. Which word embeddings to use? – IFT6266 – H2017 DEEP LEARNING
31. Word Embeddings – Piano finish standard
32. Using word embedding to enable semantic queries on relational databases | the morning paper
33. Word vectors for non-NLP data and research people – data flume.
34. Topic Classification – Bridging Topic Modelling and Text Classification – UnderstandLing Blog
35. The (Non-)Sense of Word Vectors (1/2) – UnderstandLing Blog
36. 150 多个 ML、NLP 和 Python 相关的教程 | Hello word !
37. Accelerating innovation through analogy mining | the morning paper
38. Struc2vec: learning node representations from structural identity | the morning paper
39. Chamber of Secrets: Teaching a Machine What Congress Cares About | The Reader Magazine
40. Chamber of Secrets: Teaching a Machine What Congress Cares About | Radio Free
41. Fascinating application: Teaching a Machine What Members of Congress Care About | Later On
42. Machine Learning Overview – Machine Learning for Mathies
43. Teaching Machines to Read Emails: Feature Selection – Stacks & Q's
44. 深度学习和自然语言处理：诠释词向量的魅力 | 神刀安全网
45. Convolutional neural networks for text classification – Artificial intelligence and parrots

46. 150 多个 ML、NLP 和 Python 相关的教程 - 易可信blog系统
47. How intent classification works - Mr. Bot
48. Understanding intents classification - Mr. Bot
49. Episode 3: Word Embeddings – New Paradigm
50. Building A Basic Fatwa Chat Bot - Fine Tuned
51. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | TechCrunch
52. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – The Huffington Global
53. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | Aquaiver IT Solutions (Pvt) Ltd
54. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – Cheyyur
55. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Technology NEWS
56. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – التكنولوجيا – شبكة الموقع الاول
57. How Facebook's new way of classifying what you write may speed feature rollouts across the globe | Bakhabar
58. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | KWOTABLE
59. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - DailyScene.com |
60. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Data Science Tidings
61. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – Snoopy Tech
62. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | techdom.tk
63. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe
64. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – My Cell Phone Blog
65. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | Techlear
66. How Facebook's new way of classifying what you write may speed feature rollouts across the globe | ADMK Agency
67. How Facebook's new way of classifying what you write may speed feature rollouts across the globe | News and Stuff
68. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – MagicKafe.com
69. How Facebook's new way of classifying what you write may speed feature rollouts across the globe | GREENGROUND IT
70. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Versatile Blogger
71. How Facebook's new way of classifying what you write may speed feature rollouts across the globe | Indiblog
72. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | BullViral.com
73. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – InterNetWork .OnLine

74. How Facebook's new way of classifying what you write may speed feature rollouts across the globe | Hash Virals
75. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Tech News Hub
76. How Facebook's new way of classifying what you write may speed feature rollouts across the globe | Viral Buff
77. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Domains4Change
78. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Cosmo Peek
79. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – TheTechWorld
80. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Tech Scenes
81. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Tech News Finder
82. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe > FutureTechnologyNews 2018
83. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – Inix Zone
84. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | My Tech News Today
85. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | Future Technology News 2018
86. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Digit Cool
87. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Tech News Project
88. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Technewssites
89. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe » MyTechNewsToday
90. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | CafeNews
91. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe ⋆ The Tech World
92. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Tech World
93. “BOOKr” — How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | BOOKr .. – BOOKr
94. How Facebook's new way of classifying what you write may speed feature rollouts across the globe | Digitpol
95. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – The Engineering of Conscious Experience
96. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - WebDnet
97. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - GAS NEWS
98. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe – Kopitiam Bot

99. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe - Tech life
00. How Facebooks new way of classifying what you write may speed feature rollouts across the globe - Packaging Design Services - Powered by InterDigitel
01. How Facebooks new way of classifying what you write may speed feature rollouts across the globe | TECHNEWZZ
02. How Facebook’s new way of classifying what you write may speed feature rollouts across the globe | Dawnhost
03. Learning meaningful location embeddings from unlabeled visits | Sentiance
04. 干货 | 请收下这份2018学习清单：150个最好的机器学习，NLP和Python教程-时讯快报
05. Get Busy with Word Embeddings – An Introduction | Shane Lynn
06. Dynamic word embeddings for evolving semantic discovery | the morning paper
07. Word Vectors: The Foundation of Natural Language Processing (NLP)
08. Vecteurs de mots et traitement automatique du langage naturel (TALN) | Master CAWEB
09. Game of missuggestions: semantic analysis of search autocomplete manipulation | the morning paper
10. word2vec词向量训练及gensim的使用 – Represent
11. 150 多个 ML、NLP 和 Python 相关的教程 - CAASLGlobal
12. Understanding Word Embeddings – Towards Machine Learning
13. Deep code search | the morning paper
14. Hexbyte Hacker News Computers Deep code search | HexByte Inc.
15. 自然語言處理 (NLP)：斷開中文的鎖鍊！
16. 如何斷開中文峰峰相連的詞彙鎖鍊，讓電腦能讀懂字裡行間的語意？ - PanSci 泛科學
17. 如何斷開中文峰峰相連的詞彙鎖鍊，讓電腦能讀懂字裡行間的語意？ – My Blog
18. 如何斷開中文峰峰相連的詞彙鎖鍊，讓電腦能讀懂字裡行間的語意？ - dropBlog
19. Getting started with Word Vectors – Site Title
20. Over 200 of the Best Machine Learning, NLP, and Python Tutorials — 2018 Edition - Coding Videos
21. Data Science Resources – TheMenYouWantToBe
22. Data Science Resources – Mohit Sharma (TheMenYouWantToBe) – Medium - Coding Videos
23. Artificial Intelligence: Links And Resources (81) | Angel ”Java” Lopez on Blog
24. CDL2018
25. Semantics, meet Data Science: GraphDB adds support for data wrangling and similarity search – CDL2018