Have we cracked semantics?

A practitioner's exploration into what’s possible.

Data Science Portugal Meetup (DSPT) \#8

Daniel Loureiro (danlou.github.io)

Braga, April 5th 2017

Plan for this talk

1. Why are Word Embeddings Relevant?
2. Key Concept 1: Context Windows
3. Key Concept 2: Two Shallow NN Approaches
4. Selecting a Corpus
5. Training a Word Model
6. Computing Similarities
7. Looking up Similar Words
8. Reasoning by Analogy
9. Evaluating - Analogies
10. Visualizing the Vector Space
11. Recommendation with doc2vec - Related Pages
12. Search with Skip-Thought Vectors - Similar Sentences

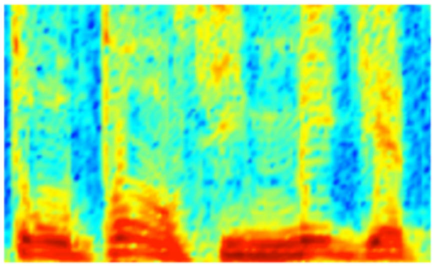
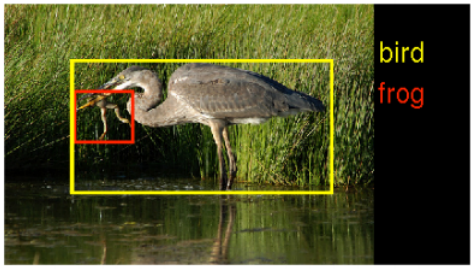
Dependencies

- Python 3.5.x
- numpy
- sklearn (0.18.1)
- scipy
- bokeh
    - I recommend you use the Anaconda distribution of Python (4.2.x) (https://repo.continuum.io/archive/index.html) which includes all of the above
- gensim (1.0.1)
- tensorflow (1.0.1)

A bit about my path

- *2011* - BIC at GECAD on Sentiment Analysis, EPIA
- *2012* - Startup Pirates, Mini Seedcamp with AskPepito
- *2013* - BsC Computer Science from FCUP, LxMLS
- *2014* - Started PepFeed, incubated Startup Braga
- *2015* - Raised seed round for PepFeed, more CEO/CTO
- *2016* - Joined Followprice, back to full-time DS
- *2017* - Left Followprice, TBD

Why are Word Embeddings Relevant?

AUDIO	IMAGES	TEXT
		
Audio Spectrogram	Image pixels	Word, context, or document vectors
DENSE	DENSE	SPARSE

Source: TensorFlow

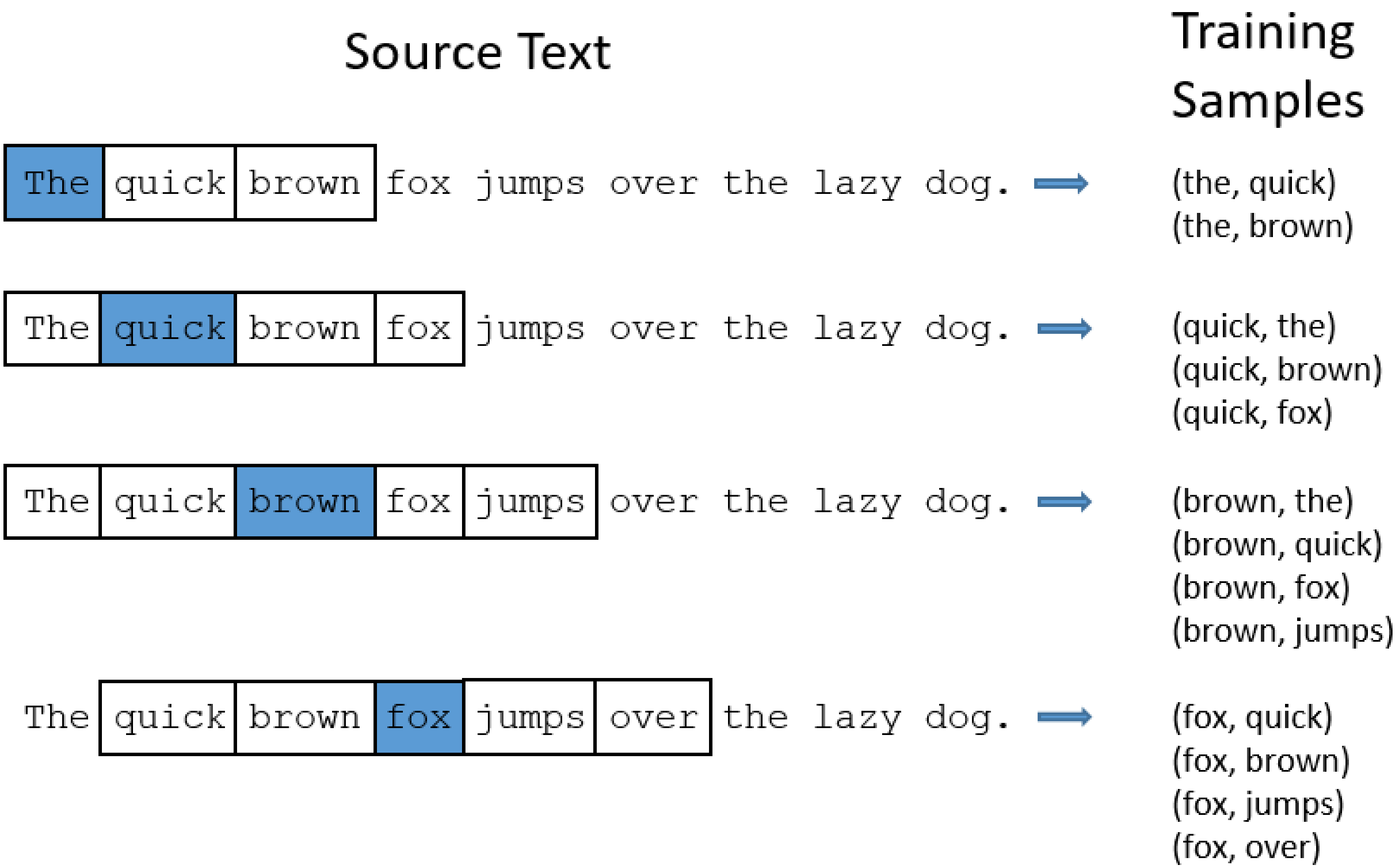
> The use of word representations... has become a "secret sauce" for the success of many NLP systems in recent years, across tasks including named entity recognition, part-of-speech tagging, parsing, and semantic role labeling.

Luong, Socher, Manning (2013) (https://nlp.stanford.edu/~lmthang/data/papers/conll13_morpho.pdf)

Context Windows

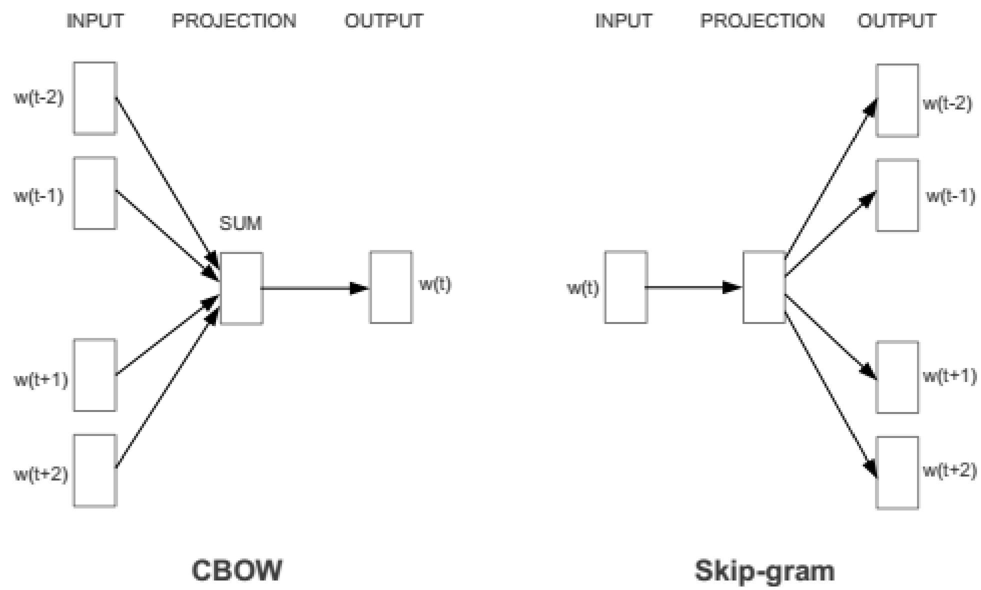
"You shall know a word by the company it keeps" (Firth, 1957)

</center>



Two Shallow NN Approaches

\



Selecting a Corpus

Marvel Wikia Corpus (http://marvel.wikia.com (http://marvel.wikia.com))

- Brand new, compiled for this DSPT session
- 27,045 text documents with details about characters, items, locations, etc. (one file)
- Demonstrate that word vectors can be useful with smaller corpuses (<< 1B tokens)
- Pre-processed - tokenized and joined multi-word expression
- Released under CC BY-NC-SA 4.0 license (free of infringements)
- Code for compiling this corpus - https://gist.github.com/danlou/532f761b6f568e20ee10a613aacea716 (https://gist.github.com/danlou/532f761b6f568e20ee10a613aacea716)

In [1]:
```
!tail -50 corpus/marvel.txt # blank line separates documents, docs are composed of article_id, title, sentences.
```

As a result Zzzax was grounded and its field was disrupted .
Somehow Zzzax was imprisoned by S.H.I.E.L.D (Earth-616) soon after Hawkeye and Wonder_Man defeated it .
SHIELD kept Zzzax , which was still in humanoid form , confined within a large insulated vacuum tube , and then transferred it into another such tube at Gamma_Base , New_Mexico .
There former General `` Thunderbolt '' Ross , who was obsessed with a desire to destroy the_Hulk , submitted to a SHIELD experiment to transform him into a superhuman being by infusing some of Zzzax 's `` living electricity '' into Ross 's body .
But the experiment went awry , and Ross 's psychic_energy -- his mind , in effect -- was absorbed by Zzzax , and Zzzax broke free .
But , strangely , Ross 's mind , perhaps because of the strength of its hatred for the_Hulk , took control of Zzzax , submerging Zzzax 's own personality .
Meanwhile , Ross ' original physical body remained alive , but all of that body 's independent thought processes had ceased ; only involuntary functions of the nervous system (such as those controlling the heartbeat) continued .
Controlled by Ross 's thought patterns , Zzzax tried unsuccessfully to kill both Bruce_Banner and Rick_Jones , who had recently become a Hulk-like monster himself .
Zzzax 's own consciousness resurfaced briefly , and fought for dominance of its physical form with the Ross persona , but the Ross consciousness again took control .
The Ross persona had apparently become unbalanced by its absorption into Zzzax , and when Banner forced it to recognize that it inhabited a_monster 's form , the Ross persona was horrified .
Still controlled by the Ross consciousness , Zzzax fled .
Eventually , Ross 's mind returned to his original physical body , bringing much of Zzzax 's living electricity with it .
But now a new menace had come to Gamma_Base : a grotesque mutant (The_Nevermind) , which seized upon the heads of humans_victims and drained life-energy from their bodies , killing them .
Ross watched as Rick_Jones , in his Hulk-like form , tried unsuccessfully to defeat the mutant , which menaced Betty .
Then , to save Betty , Bruce_Banner hurled himself into the mutant 's path , and thereby himself became the mutant 's victim .
On seeing the_Hulk 's bravery and Bruce_Banner 's self-sacrifice , Ross realized that he had gravely misjudged them both .
The mutant rendered Bruce_Banner unconscious without draining him of all his life energy , and then headed towards Betty .
This time , Thaddeus_Ross hurled himself into the mutant 's path , and the creature took hold of his head .
But since Ross still possessed much of Zzzax 's electrical energy , the mutant could not take control of Ross 's mind , and Ross released all of Zzzax 's energy , electrocuting the mutant .
Zzzax has since returned , however .
Zzzax was an `` electromagnetic intelligence , '' a psionically-charged electromagnetic field which had a_humanoid_form and which was capable of crude human level intelligence and superhuman_strength .
Zzzax generally appeared as a gargantuan mass of electrical `` sparks '' in humanoid form .

It was highly luminescent , whitish-yellow in color , and gave off the smell of ozone .
Zzzax could hover or fly in the air .
Zzzax could fire extraordinarily powerful blasts of electricity .
Zzzax 's tremendously high voltage enabled it to incinerate matter which lied in its path .
Zzzax 's superhuman_strength enabled him to battle the_Hulk nearly to a standstill .
Zzzax incinerated whole city blocks in minutes .
Zzzax burned the_Hulk 's skin to a slight degree .
At its most powerful , Zzzax was forty feet tall and carried a voltage of several hundred thousand volts that could be almost totally expended over a twenty minute period .
Zzzax subsisted by drawing all available electromagnetic fields into its own .
Those electromagnetic fields housed by human brains , when absorbed , added to Zzzax 's level of stored psionic energy , making it more intelligent .
For an unknown reason Zzzax found this absorption of electrical/psionic energy from human brains pleasurable , and therefore undertook the murder of human_beings for their psionic energy whenever possible , In absorbing this psionic energy from a human victim , Zzzax would incinerate the victim 's body , Zzzax was seemingly unable to absorb psionic energy from its recurring adversary , the_Hulk .
However , Zzzax could take control of the electrical/psionic impulses in the_Hulk 's nervous system , and thereby control the activities of the_Hulk 's muscles .
Zzzax 's own thoughts and behavior could be influenced by strong desires on the part of the human_beings whose psionic energy it had consumed .
However , only in the case of Thunderbolt_Ross had another_being 's consciousness supplanted Zzzax 's own as the dominant one controlling Zzzax 's physical form Electricity Physiology : Zzzax is a creature of pure electricity (or to be more precise a psionically charged electromagnetic field in humanoid form) .
It can generate electricity (usually in the form of lightning bolts) , manipulate nearby electrical fields , and fly .
Zzzax absorbs electricity from the human brain to survive ; this usually kills its victims , and usually gives Zzzax temporary personality traits similar to those of the_person it has absorbed .
Only his foe the_Hulk has proven immune to this ability .
Although Zzzax is a being composed of energy , what passes as its corporeal form is able to lift matter as a normal humanoid body (as though it had hands , muscles , and a skeletal structure) .
Zzzax possesses the ability to lift in excess of 100_tons or more based on its level of energy it has absorbed at the time .
Zzzax has nearly limitless stamina and durability due to the quasi-physical nature of its body .
It is capable of hovering its body in the air , but it does not achieve a velocity any faster than that of an ordinary human of its size who was running or walking at the same speed on level ground .
When the energy that composes Zzzax 's quasi-physical body is dispersed or nullified it somehow always reforms itself .
Zzzax is extremely difficult to restrain or damage .

674537

```
Äkräs
Äkräs served as the champion of mortals and protector of their crops .
```

In [2]:

```python
# let's load the corpus
from collections import OrderedDict
marvel_articles = OrderedDict()

article_lines = []
with open('corpus/marvel.txt', encoding='utf-8') as f:
    for line in f:
        line = line.strip()

        if len(line) > 0:
            article_lines.append(line)
        else:
            article_id = int(article_lines[0])
            article_name = article_lines[1]
            article_text = article_lines[2:]
            marvel_articles[article_id] = dict(name=article_id, text=article_text)

            article_lines = []  # reset for next article/document
```

In [3]:

```python
len(marvel_articles)
```

Out[3]:

```
27045
```

In [4]:
```
tony_stark = marvel_articles[1868]
print(tony_stark['text'][:25])
print(len(tony_stark['text']))
```

['The biological parents of Tony_Stark were two S.H.I.E.L.D agents , Amanda_Armstrong and Jude , who met during a courier mission .', 'After Jude saved Amanda from an assassin , they got to know each other and fell in love .', 'Following a two-year relationship , Amanda became pregnant .', 'A week before giving birth to the baby , Jude revealed to have been a Hydra double-agent with little regard for anybody but Amanda and himself who sold out fellow S.H.I.E.L.D soldiers , and was even responsible for the incident that had almost cost Amanda her life .', "During a discussion when he was trying to convince Amanda to accept Hydra 's protection , she attacked Jude and killed him .", 'Traumatized by this development , Amanda asked S.H.I.E.L.D to ensure her future baby would find a safe and happy home .', 'However , director Nick_Fury followed the same procedure used for unwanted pregnancies in the agency , and the baby was left in an orphanage in Sofia , Bulgaria after Amanda birthed him in a local hospital .', "Fury 's associate and famous industrialist Howard_Stark learned of this , and decided to find the baby and adopt him , keeping the name Amanda wished he retained : Anthony .", "In addition to Howard and his_wife Maria suffering the latter 's inability to give birth again , they needed to find a healthy boy to act as a decoy in place of their secret first born , Arno_Stark .", "Arno 's gestation had been extremely difficult , and his birth was only made possible with the help of an alien robot , the Rigellian_Recorder 451 , who had agreed to help the baby survive in exchange of the opportunity to bio-engineer him , so he could accelerate humanity 's technological growth in the_future .", "However , as 451 genetically modified the baby in womb , Howard had discovered the robot hid some sort of kill switch , that would compromise the life of his_son in the_future , for which Stark developed a `` biococktail '' to interfere with it behind 451 's back .", 'Once Arno was born , 451 left the Earth .', "In a turn of events , Howard 's interference with 451 's machinations had caused the newborn to become fatally ill .", 'The Starks had decided to keep the baby hidden in the Maria_Stark_Foundation Hospice .', "In addition to filling the void left by Arno 's fatal illness , Tony 's adoption would prevent 451 from learning of Howard 's meddling if it ever returned to the Earth .", "Tony grew up completely unaware of Arno 's existence or that he was adopted .", "While loved unconditionally by Maria , Tony suffered from a strained relationship with his_father , both due to the constrast of Tony 's sensitive and reclusive nature with Howard 's glorification of physical prowess and Howard 's ever-increasing drinking habits , which caused him to verbally abuse Tony and suffer from mood swings .", 'This last factor caused Tony to turn to electronics as a coping mechanism at barely five years old , as he started to believe hardware to be comprehensible and reliable , whereas people were unpredictable and hard to understand .', "Tony 's world could n't find order , but the things he built did .", "In order to toughen his_son , Howard sent Tony to boarding school at the age of seven , much to Maria 's dismay .", 'The following years , Tony learned of discipline of body and strength of character as Howard intended , while spending his free time reading alone .', 'At the age of thirteen , the stories of Thomas_Malory opened Tony the doors to a new world of dedication to a cause greater than oneself , of chivalry , honor , and armored heroes .', 'After boarding school , Tony joined an undergraduate program at MIT at the age of fifteen .', '[verification needed] He would effortlessly graduate as class valedictorian with double majors in physics and engineering .', "When he was seventeen , Tony met Meredith_McCall , his first love and , unfortunately , the daughter of Howard 's greatest business rival ."]
452

Training a Word Model

We'll be using the highly praised gensim (https://github.com/RaRe-Technologies/gensim) package and their implementation of word2vec.

In [5]:
```
# we need to isolate the sentence tokens from the corpus
marvel_sents = [sent for e in marvel_articles.values() for sent in e['text']]
marvel_sents = [sent.lower().split() for sent in marvel_sents]
len([w for sent in marvel_sents for w in sent]) # our total number of tokens, 6M, very small
```

Out[5]:
```
6226813
```

In [6]:
```
# we also want to know the number of cores on the machine to take advantage of multithreading
import multiprocessing
cores = multiprocessing.cpu_count()
```

In [7]:
```
from gensim.models import Word2Vec

# creating models will always take a few minutes, load pretrained whenever available
import os.path
if os.path.isfile('pretrained/w2v_sg_marvel'):
    word_model = Word2Vec.load('pretrained/w2v_sg_marvel')

else:
    # params are standard 300 dimensions and flag to use skip-gram instead of cbow
    word_model = Word2Vec(marvel_sents, size=300, min_count=2, sg=1, workers=cores)
    word_model.save('pretrained/w2v_sg_marvel')
```

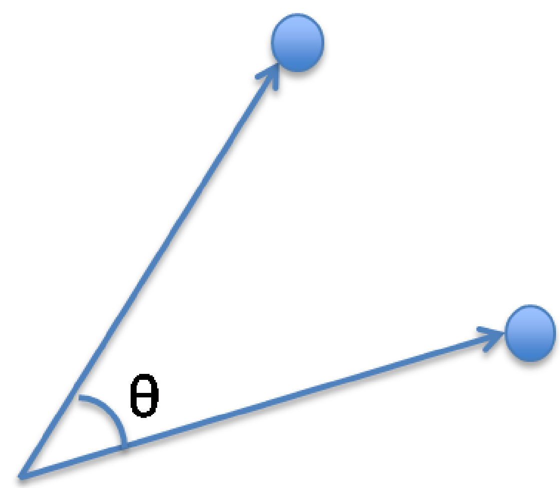
In [8]:
```
# voila, we have word vectors!
word_model.wv['iron_man']
```

```
Out[8]: array([ 0.23367834,  0.29375732, -0.45883524, -0.17062214,  0.06527832,
                0.03812009, -0.295131  ,  0.01019869, -0.12619233, -0.04480179,
               -0.59440571,  0.31791714, -0.28028834, -0.23229396,  0.13952047,
               -0.31170291, -0.27238059, -0.07909798,  0.20126045, -0.08187994,
                0.00599575, -0.1405274 , -0.11241629,  0.16259389, -0.20842955,
               -0.05150102,  0.13531914, -0.14348233, -0.33023039, -0.30291417,
               -0.61079025,  0.17066939,  0.26637861, -0.09950258,  0.12790087,
                0.37538701, -0.35994497,  0.20974636,  0.00361202,  0.20891951,
               -0.16384113, -0.22473757,  0.07013547, -0.09483541,  0.05740248,
                0.09562255,  0.31106865,  0.02298987, -0.01928948, -0.00395847,
               -0.1754694 ,  0.34890848, -0.06538704, -0.42149439, -0.29546908,
                0.00601043,  0.0114274 , -0.18908688,  0.01014967, -0.33985901,
               -0.18500525, -0.84454107, -0.11101522, -0.30179295,  0.06611467,
                0.07937766, -0.03002396, -0.2946749 , -0.0373145 , -0.11919513,
               -0.13816983,  0.11353034, -0.05764449,  0.31857526, -0.14565814,
                0.22176565, -0.0092652 , -0.12686509, -0.04012004,  0.34117752,
               -0.19678129, -0.15876527, -0.33362925, -0.2795046 ,  0.09368271,
               -0.57504612, -0.01154695,  0.09723811,  0.06061403, -0.35488963,
                0.18581401,  0.00658177, -0.02415015, -0.03357187, -0.0720716 ,
                0.25740573,  0.13612902, -0.1162134 , -0.02310944, -0.05195754,
                0.67955214, -0.2812956 ,  0.23084752, -0.05945769, -0.01722902,
                0.17378171,  0.27284735,  0.03824491, -0.41214183,  0.34434503,
               -0.06845174, -0.0403631 ,  0.25560844, -0.29991204, -0.06057943,
               -0.21231087, -0.22515942, -0.08749398,  0.13168706,  0.01142654,
                0.10096975, -0.06992551, -0.19930084,  0.0551247 ,  0.36017701,
               -0.03313124,  0.14378056,  0.53382754,  0.0722019 , -0.11850061,
               -0.08871941,  0.23279266,  0.31996784,  0.08891867, -0.01219382,
               -0.2321573 , -0.12022872,  0.09479933,  0.21503246, -0.29450843,
                0.05400112, -0.36321825,  0.2556532 ,  0.1256983 ,  0.12850764,
               -0.10713939,  0.053009  , -0.20115253, -0.30178368, -0.26040792,
                0.30832961,  0.16036382, -0.16917124,  0.05111419,  0.3465938 ,
                0.02046909,  0.33141509,  0.42179662,  0.01219467, -0.36196905,
               -0.33664575, -0.06266017, -0.32503498,  0.39983475,  0.07438868,
               -0.15231924,  0.20126805, -0.03444426, -0.28328151, -0.20903379,
                0.14077148,  0.24051175, -0.29495713, -0.28576964, -0.19436385,
               -0.18110064,  0.12513387, -0.2558167 ,  0.005901  , -0.15428029,
                0.025522  , -0.41709223, -0.05957717,  0.10703159,  0.10984971,
                0.15024401,  0.21283416,  0.28829446,  0.25687751,  0.38858017,
                0.14893845, -0.08103204,  0.13624971,  0.41810927, -0.31016475,
                0.12883775, -0.12224952,  0.04709034,  0.0730826 ,  0.09438442,
               -0.25932485, -0.11641676, -0.08620116, -0.03643035,  0.34107041,
                0.2661908 , -0.28551576,  0.20380901, -0.08238478,  0.16725872,
               -0.06237131,  0.58707136,  0.1258394 , -0.03471827, -0.74544215,
```

```
-0.11081678, -0.24844058,  0.28623778, -0.05104716, -0.29103458,
 0.44071785, -0.11948644,  0.13047318,  0.12539124,  0.15627515,
-0.0129627 , -0.21479344, -0.37842533, -0.1824726 ,  0.39061427,
 0.41645542, -0.04729101, -0.37454954,  0.07649589,  0.00551359,
 0.15709005, -0.09826116, -0.04467909, -0.13486096,  0.10237116,
-0.11485546, -0.07020838,  0.10665817,  0.16139315,  0.01256329,
-0.01741873,  0.05980036,  0.07975876, -0.34957972, -0.07939122,
-0.09496113,  0.12674569,  0.15406217, -0.1403691 , -0.08632806,
 0.31916255,  0.10138023,  0.02705557, -0.61672413,  0.14203289,
 0.1719941 , -0.14900905,  0.27612299, -0.40663832,  0.19826645,
 0.13653916,  0.21502608, -0.18318695,  0.28791147,  0.05835493,
 0.13311645, -0.22190996, -0.13391089, -0.34238729,  0.33309025,
-0.15416151,  0.33515731, -0.25248697,  0.04696179, -0.07519623,
 0.42544293, -0.80132371, -0.07650273,  0.00496452,  0.07345594,
 0.02998696, -0.06085769, -0.06669183, -0.04240202, -0.12938733,
 0.2172662 , -0.24730772,  0.2212265 , -0.25555411, -0.35746542,
 0.21847862, -0.14920726,  0.06916033,  0.19325928,  0.2781345 ], dtype=float32)
```

Computing Similarities

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



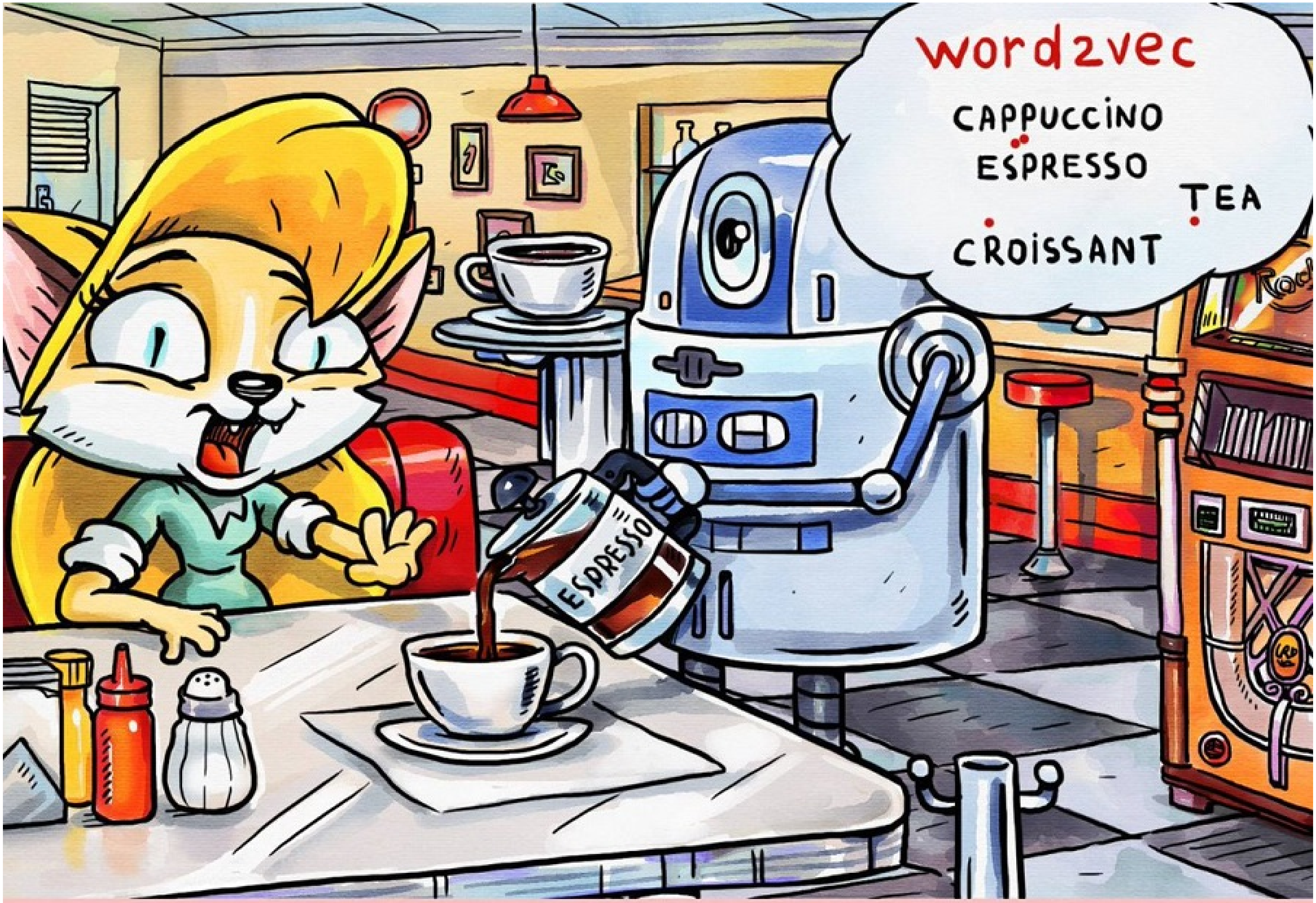
In [9]:
```
# let's compute some similarities then
import numpy as np

v1 = word_model.wv['iron_man']
v2 = word_model.wv['tony_stark']
v3 = word_model.wv['matt_murdock']

print(np.dot(v1, v2)/(np.linalg.norm(v1)*np.linalg.norm(v2)))
print(np.dot(v1, v3)/(np.linalg.norm(v1)*np.linalg.norm(v3)))
```

```
0.701744
0.408435
```

In [10]:
```
# or more simply using gensim
word_model.init_sims()  # making sure we've computed norms, may be unnecessary
print(word_model.similarity('iron_man', 'tony_stark'))
print(word_model.similarity('iron_man', 'matt_murdock'))
```

```
0.701743639756
0.408435017743
```

Looking up Similar Words



- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing

that they are almost the same thing.

In [11]:
```
# fetch the normalized vector (simplifies computation)
w1_idx = word_model.wv.index2word.index('avengers')
v1_norm = word_model.wv.syn0norm[w1_idx]

sims = np.dot(word_model.wv.syn0norm, v1_norm)  # cosine sims for ALL vecs
sims = [word_model.wv.index2word[idx] for idx in np.argsort(sims)]  # corresponding words
sims = sims[::-1]  # reverse the list (lower is best)
sims[:10]
```

Out[11]:
```
['avengers',
 'the_avengers',
 'mighty_avengers',
 'avengers_team',
 'secret_avengers',
 'force_works',
 'new_avengers',
 'unity_division',
 'west_coast_avengers',
 'x-men']
```

In [12]:
```
# or, once more, with gensim
word_model.most_similar('avengers', topn=10)
```

Out[12]:
```
[('the_avengers', 0.700478196144104),
 ('mighty_avengers', 0.6753344535827637),
 ('avengers_team', 0.6537984609603882),
 ('secret_avengers', 0.65278559923172),
 ('force_works', 0.647817850112915),
 ('new_avengers', 0.6450065970420837),
 ('unity_division', 0.6299926042556763),
 ('west_coast_avengers', 0.6266465187072754),
 ('x-men', 0.6255587339401245),
 ('avengers_west_coast', 0.6206583976745605)]
```

Reasoning by Analogy

In [13]:
```python
# can we discover the true identity of super-heroes?
identities = [('iron_man', 'tony_stark'), ('the_hulk', 'bruce_banner'), ('captain_america', 'steve_rogers'), ('falcon', 'sam_wilson')]

def normvec(label):  # aux method
    return word_model.wv.syn0norm[word_model.wv.index2word.index(label)]

v = normvec('tony_stark') - normvec('iron_man') + normvec('captain_america')

word_model.similar_by_vector(v)
```

Out[13]:
```
[('captain_america', 0.772327184677124),
 ('rogers', 0.56960129737854),
 ('steve_rogers', 0.5688270330429077),
 ('cap', 0.562447190284729),
 ('tony_stark', 0.5591169595718384),
 ('sharon_carter', 0.5284056663513184),
 ('bucky_barnes', 0.5162870287895203),
 ('james_barnes', 0.504021167755127),
 ('barnes', 0.4948274791240692),
 ('bucky', 0.4910045266151428)]
```

In [14]:
```python
# the equivalent with gensim
word_model.most_similar(positive=['tony_stark', 'captain_america'], negative=['iron_man'])
```

Out[14]:
```
[('rogers', 0.56960129737854),
 ('steve_rogers', 0.5688270330429077),
 ('cap', 0.562447190284729),
 ('sharon_carter', 0.5284056663513184),
 ('bucky_barnes', 0.5162870287895203),
 ('james_barnes', 0.504021167755127),
 ('barnes', 0.4948274791240692),
 ('bucky', 0.4910045564174652),
 ('nick_fury', 0.489604651927948),
 ('the_red_skull', 0.4890736937522888)]
```

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

Evaluating - Analogies (syntactic and semantic)

In [15]:
```
import logging
logging.basicConfig(level=logging.INFO)
results = word_model.accuracy('eval/questions-words.txt')
```

```
INFO:gensim.models.keyedvectors:capital-common-countries: 10.4% (19/182)
INFO:gensim.models.keyedvectors:capital-world: 5.9% (9/152)
INFO:gensim.models.keyedvectors:currency: 0.0% (0/28)
INFO:gensim.models.keyedvectors:city-in-state: 3.3% (18/544)
INFO:gensim.models.keyedvectors:family: 38.6% (132/342)
INFO:gensim.models.keyedvectors:gram1-adjective-to-adverb: 2.3% (20/870)
INFO:gensim.models.keyedvectors:gram2-opposite: 0.5% (1/210)
INFO:gensim.models.keyedvectors:gram3-comparative: 16.3% (123/756)
INFO:gensim.models.keyedvectors:gram4-superlative: 8.3% (20/240)
INFO:gensim.models.keyedvectors:gram5-present-participle: 13.7% (111/812)
INFO:gensim.models.keyedvectors:gram6-nationality-adjective: 3.0% (19/633)
INFO:gensim.models.keyedvectors:gram7-past-tense: 12.6% (187/1482)
INFO:gensim.models.keyedvectors:gram8-plural: 11.7% (102/870)
INFO:gensim.models.keyedvectors:gram9-plural-verbs: 15.8% (60/380)
INFO:gensim.models.keyedvectors:total: 10.9% (821/7501)
```

Not so good... maybe we can do better with more data

Expanding Vector Space

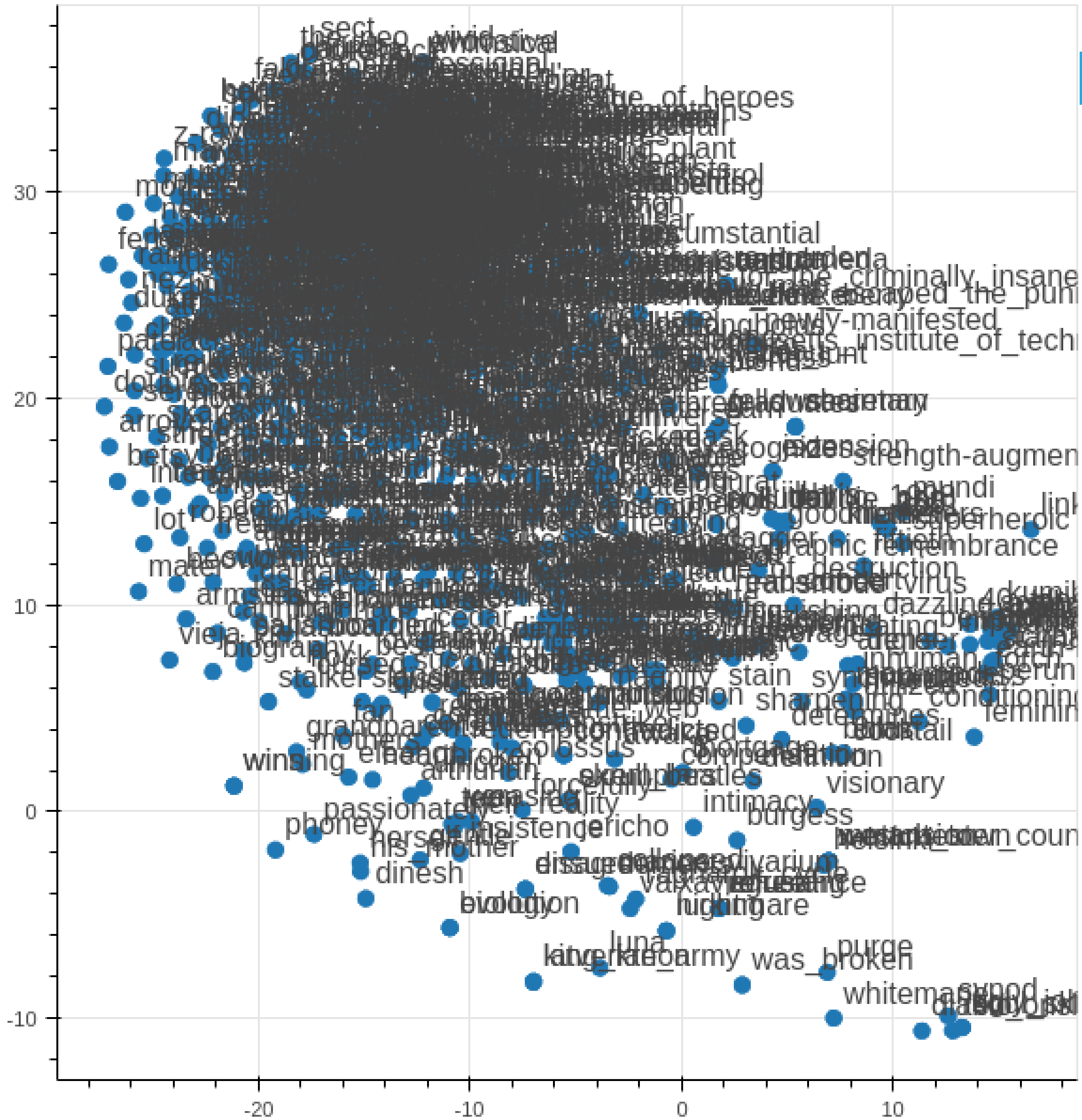
In [16]:
```python
# the text8 corpus is 100MB of tokenized text from wikipedia (N tokens)
from gensim.models.word2vec import Text8Corpus

if os.path.isfile('pretrained/w2v_sg_marvel_text8'):
    # load from pretrained
    word_model = Word2Vec.load('pretrained/w2v_sg_marvel_text8')

else:
    # continue training
    word_model.train(Text8Corpus('corpus/text8'))
    word_model.save('pretrained/w2v_sg_marvel_text8')

results = word_model.accuracy('eval/questions-words.txt')
```

```
INFO:gensim.utils:loading Word2Vec object from pretrained/w2v_sg_marvel_text8
INFO:gensim.utils:loading wv recursively from pretrained/w2v_sg_marvel_text8.wv.* with mmap=None
INFO:gensim.utils:loading syn0 from pretrained/w2v_sg_marvel_text8.wv.syn0.npy with mmap=None
INFO:gensim.utils:setting ignored attribute syn0norm to None
INFO:gensim.utils:loading syn1neg from pretrained/w2v_sg_marvel_text8.syn1neg.npy with mmap=None
INFO:gensim.utils:setting ignored attribute cum_table to None
INFO:gensim.utils:loaded pretrained/w2v_sg_marvel_text8
INFO:gensim.models.keyedvectors:precomputing L2-norms of word weight vectors
INFO:gensim.models.keyedvectors:capital-common-countries: 35.9% (56/156)
INFO:gensim.models.keyedvectors:capital-world: 28.1% (39/139)
INFO:gensim.models.keyedvectors:currency: 0.0% (0/18)
INFO:gensim.models.keyedvectors:city-in-state: 18.8% (108/576)
INFO:gensim.models.keyedvectors:family: 32.5% (111/342)
INFO:gensim.models.keyedvectors:gram1-adjective-to-adverb: 3.1% (27/870)
INFO:gensim.models.keyedvectors:gram2-opposite: 8.8% (16/182)
INFO:gensim.models.keyedvectors:gram3-comparative: 34.5% (261/756)
INFO:gensim.models.keyedvectors:gram4-superlative: 22.1% (53/240)
INFO:gensim.models.keyedvectors:gram5-present-participle: 7.0% (57/812)
INFO:gensim.models.keyedvectors:gram6-nationality-adjective: 60.2% (381/633)
INFO:gensim.models.keyedvectors:gram7-past-tense: 15.5% (229/1482)
INFO:gensim.models.keyedvectors:gram8-plural: 33.7% (274/812)
INFO:gensim.models.keyedvectors:gram9-plural-verbs: 21.8% (83/380)
INFO:gensim.models.keyedvectors:total: 22.9% (1695/7398)
```

Visualizing the Vector Space

In [17]:
```
# select random n vectors (normalized)
from random import sample
words = sample(list(word_model.wv.vocab.keys()), 1000)
vecs = [normvec(n) for n in words]

from sklearn.manifold import TSNE
vecs_2d = TSNE(n_components=2).fit_transform(vecs)

from bokeh.plotting import figure, output_notebook, show
from bokeh.models import ColumnDataSource, LabelSet
output_notebook()

source = ColumnDataSource(data=dict(x=vecs_2d[:, 0], y=vecs_2d[:, 1], labels=words))

p = figure()
p.scatter('x', 'y', size=8, source=source)

# include labels
labels = LabelSet(x='x', y='y', text='labels', level='glyph',
                  x_offset=5, y_offset=5, source=source, render_mode='canvas')
p.add_layout(labels)

show(p)
```

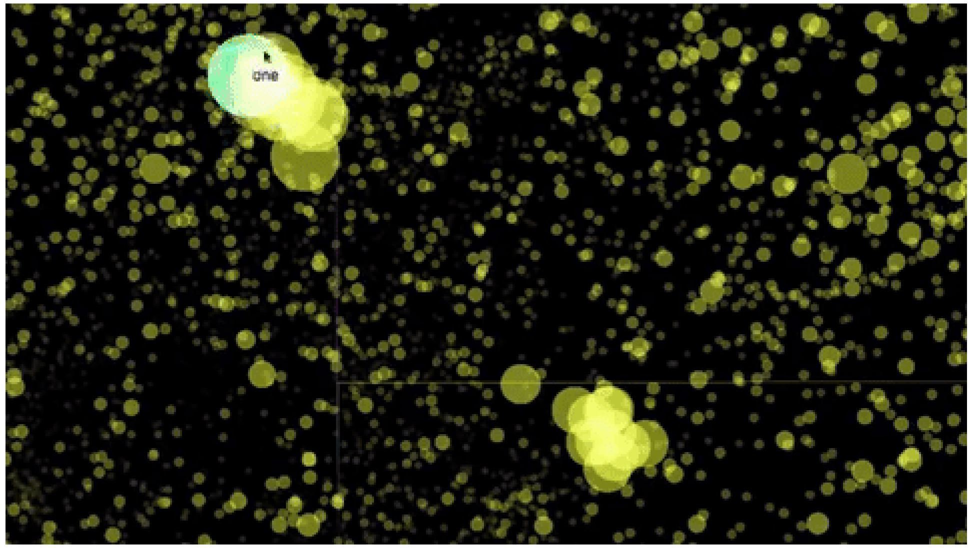
(http://bokeh.pydata.org)BokehJS successfully loaded.



(http://bokeh.pydata.org/)

Find any clusters? We may need a larger corpus for this...

TSNE with TensorFlow (TensorBoard)



[Tutorial (https://www.tensorflow.org/get_started/embedding_viz)](https://www.tensorflow.org/get_started/embedding_viz)

This concludes the overview of what's possible with dense representations of words.

Next we'll look at derivative methods for recommendation and search.

Recommendation - Related Pages

Paragraph Vectors (doc2vec)

\

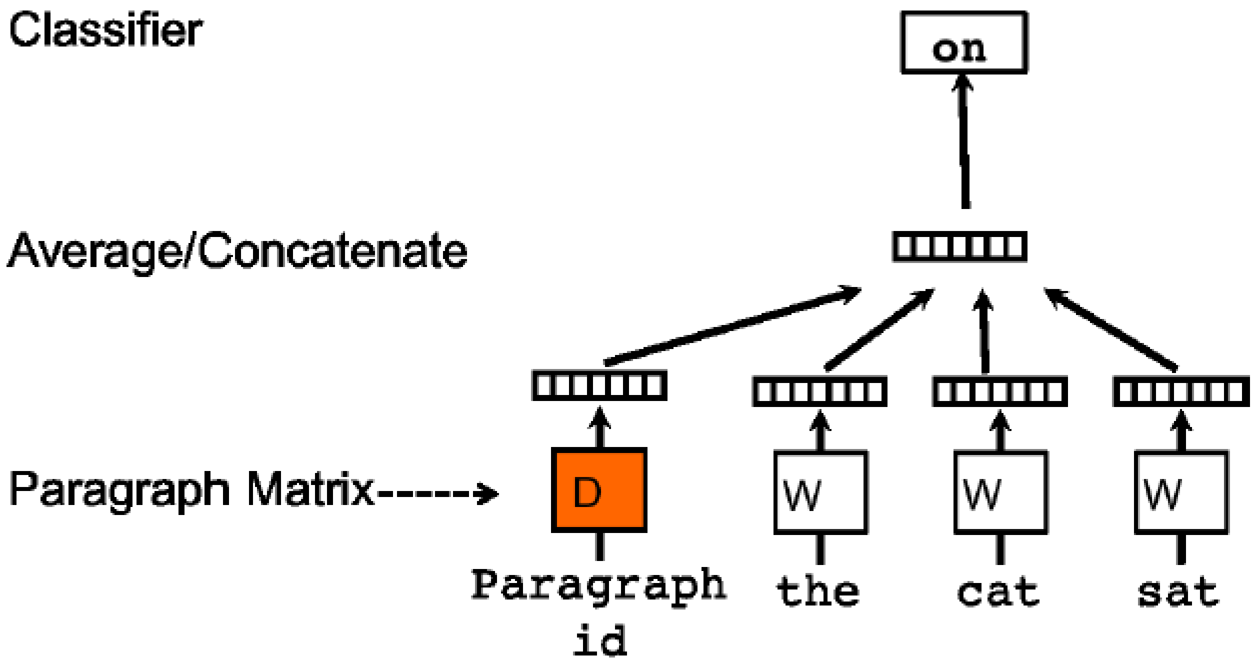


Figure 2. A framework for learning paragraph vector. This framework is similar to the framework presented in Figure 1; the only change is the additional paragraph token that is mapped to a vector via matrix D. In this model, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph.

In [18]:
```
from gensim.models.doc2vec import Doc2Vec
from collections import namedtuple

if os.path.isfile('pretrained/pv_dm_concat_marvel'):

    # load pretrained
    doc_model = Doc2Vec.load('pretrained/pv_dm_concat_marvel')

else:

    docs = []
    Document = namedtuple('Document', 'words tags id')
    for doc_id in marvel_articles:
        doc_name = marvel_articles[doc_id]['name']
        doc_text = marvel_articles[doc_id]['text']
        doc_words = [word.lower() for sent in doc_text for word in sent.split()]
        docs.append(Document(doc_words, [doc_name], doc_id))

    # PV-DM with concatenation (preserves ordering, recommended by authors)
    doc_model = Doc2Vec(dm=1, dm_concat=1, size=100, window=5, negative=5, hs=0, min_count=2, wor
kers=cores, iter=20)

    alpha, min_alpha, passes = (0.025, 0.001, 20)
    alpha_delta = (alpha - min_alpha) / passes

    for epoch in range(passes):
        shuffle(docs)  # shuffling gets best results

        doc_model.alpha, doc_model.min_alpha = alpha, alpha
        doc_model.train(docs, total_examples=doc_model.corpus_count)

        print('completed pass %i at alpha %f' % (epoch + 1, alpha))
        alpha -= alpha_delta

    doc_model.save('pretrained/pv_dm_concat_marvel')
```

```
INFO:gensim.utils:loading Doc2Vec object from pretrained/pv_dm_concat_marvel
INFO:gensim.utils:loading wv recursively from pretrained/pv_dm_concat_marvel.wv.* with mmap=None
INFO:gensim.utils:setting ignored attribute syn0norm to None
INFO:gensim.utils:loading docvecs recursively from pretrained/pv_dm_concat_marvel.docvecs.* with m
map=None
INFO:gensim.utils:loading syn1neg from pretrained/pv_dm_concat_marvel.syn1neg.npy with mmap=None
INFO:gensim.utils:setting ignored attribute cum_table to None
INFO:gensim.utils:loaded pretrained/pv_dm_concat_marvel
```

In [19]:

```
# get semantically related pages with a simple similarity lookup among all pages
# would be perfect for http://marvel.wikia.com/wiki/Peter_Parker_(Earth-616)
doc_model.docvecs.most_similar('Peter Parker')
```

```
INFO:gensim.models.doc2vec:precomputing L2-norms of doc weight vectors
```

Out[19]:

```
[('Mary Jane Watson', 0.6484307050704956),
 ('Otto Octavius', 0.6395471096038818),
 ('Doris Urich', 0.5727440118789673),
 ('May Reilly', 0.5726701021194458),
 ('Sally Green', 0.5616764426231384),
 ('Nancy Stacy', 0.5563001036643982),
 ('Kaine Parker', 0.55009925365448),
 ('Ben Reilly', 0.5442812442779541),
 ('Norman Osborn', 0.544121265411377),
 ('John Jonah Jameson', 0.5329504013061523)]
```

In [20]:

```
# another example
doc_model.docvecs.most_similar('Avengers')
```

Out[20]:

```
[('Illuminati', 0.5736404657363892),
 ('Steven Rogers', 0.5377088785171509),
 ('Avengers vs. X-Men (Event)', 0.5318738222122192),
 ('Anthony Stark', 0.5049740672111511),
 ('Time Runs Out', 0.4869905114173889),
 ('Infinity (Event)', 0.4693588614463806),
 ('Avengers (Heroes Reborn)', 0.4689474403858185),
 ('Civil War (Event)', 0.46763166785240173),
 ('Carol Danvers', 0.4660409092903137),
 ('Canarsie', 0.4616173207759857)]
```

In [21]:

```
# TODO: compare results with tf-idf
```

Search - Similar Sentences

Skip-Thought Vectors (sent2vec)

\

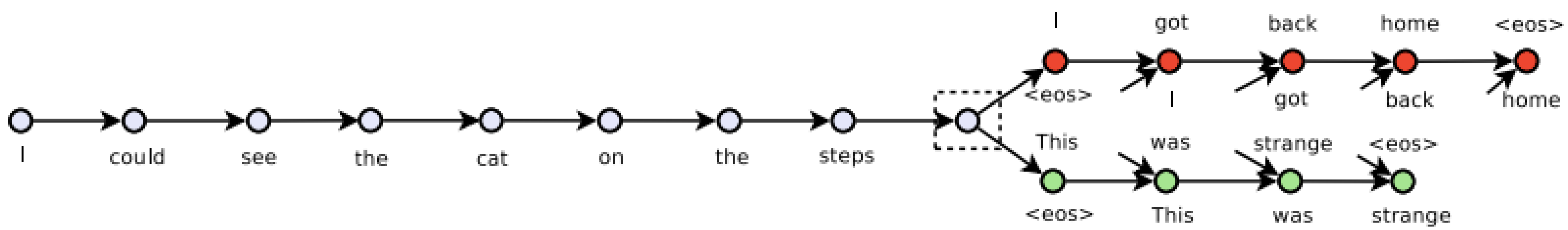


Figure 1: The skip-thoughts model. Given a tuple (s_{i-1}, s_i, s_{i+1}) of contiguous sentences, with s_i the i-th sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence s_{i-1} and next sentence s_{i+1}. In this example, the input is the sentence triplet *I got back home. I could see the cat on the steps. This was strange.* Unattached arrows are connected to the encoder output. Colors indicate which components share parameters. $\langle \text{eos} \rangle$ is the end of sentence token.

In [22]:
```
from skip_thoughts import configuration
from skip_thoughts import encoder_manager

# based on https://github.com/tensorflow/models/tree/master/skip_thoughts#encoding-sentences
PRETRAINED_UNI_DIR = 'pretrained/skip_thoughts_uni_2017_02_02/'
VOCAB_FILE = PRETRAINED_UNI_DIR+"vocab.txt"
EMBEDDING_MATRIX_FILE = PRETRAINED_UNI_DIR+"embeddings.npy"
CHECKPOINT_PATH = PRETRAINED_UNI_DIR+"model.ckpt-501424"

encoder = encoder_manager.EncoderManager()
encoder.load_model(configuration.model_config(),
                   vocabulary_file=VOCAB_FILE,
                   embedding_matrix_file=EMBEDDING_MATRIX_FILE,
                   checkpoint_path=CHECKPOINT_PATH)

# TODO: use vocabulary expansion with previously defined word_model
```

INFO:tensorflow:Reading vocabulary from pretrained/skip_thoughts_uni_2017_02_02/vocab.txt

INFO:tensorflow:Reading vocabulary from pretrained/skip_thoughts_uni_2017_02_02/vocab.txt

INFO:tensorflow:Loaded vocabulary with 930914 words.

INFO:tensorflow:Loaded vocabulary with 930914 words.

INFO:tensorflow:Loading embedding matrix from pretrained/skip_thoughts_uni_2017_02_02/embeddings.npy

INFO:tensorflow:Loading embedding matrix from pretrained/skip_thoughts_uni_2017_02_02/embeddings.npy

INFO:tensorflow:Loaded embedding matrix with shape (930914, 620)

INFO:tensorflow:Loaded embedding matrix with shape (930914, 620)

INFO:tensorflow:Building model.

INFO:tensorflow:Building model.

INFO:tensorflow:Loading model from checkpoint: pretrained/skip_thoughts_uni_2017_02_02/model.ckpt-501424

INFO:tensorflow:Loading model from checkpoint: pretrained/skip_thoughts_uni_2017_02_02/model.ckpt-501424

INFO:tensorflow:Successfully loaded checkpoint: model.ckpt-501424

INFO:tensorflow:Successfully loaded checkpoint: model.ckpt-501424

In [23]:
```
# this is a workaround for a bug in python for Mac OS X for files > 2GB
# issue: http://bugs.python.org/issue24658
# workaround: http://stackoverflow.com/a/41613221/5641547
class MacOSFile(object):
    def __init__(self, f):
        self.f = f
    def __getattr__(self, item):
        return getattr(self.f, item)
    def read(self, n):
        if n >= (1 << 31):
            buffer = bytearray(n)
            pos = 0
            while pos < n:
                size = min(n - pos, 1 << 31 - 1)
                chunk = self.f.read(size)
                buffer[pos:pos + size] = chunk
                pos += size
            return buffer
        return self.f.read(n)

import pickle
import platform

if os.path.isfile('pretrained/marvel_sent_encodings.p'):
    # this isn't as much pretrained as it's precomputed

    if platform.system() == 'Darwin':
        encodings = pickle.load(MacOSFile(open('pretrained/marvel_sent_encodings.p', 'rb')))
    else:
        encodings = pickle.load(open('pretrained/marvel_sent_encodings.p', 'rb'))

else:
    data = [' '.join(sent) for sent in marvel_sents]
    # generate skip-thought vectors for each sentence in the dataset
    encodings = encoder.encode(data)

    if platform.system() == 'Darwin':
        pickle.dump(MacOSFile(open('pretrained/marvel_sent_encodings.p', 'wb')))  #untested
    else:
        pickle.dump(open('pretrained/marvel_sent_encodings.p', 'wb'))
```

In [24]:
```
import scipy.spatial.distance as sd

# helper function to generate nearest neighbors / similar sentences.
def get_sim_sents(sent, num=10):
    encoding = encoder.encode([sent])[0]
    scores = sd.cdist([encoding], encodings, "cosine")[0]
    sorted_ids = np.argsort(scores)
    print("Sentence:")
    print("", sent)
    print("\nNearest neighbors:")
    for i in range(1, num + 1):
        print(" %d. %s (%.3f)" %
              (i, ' '.join(marvel_sents[sorted_ids[i]]), scores[sorted_ids[i]]))

# some example sentences from the corpus
get_sim_sents('professor gilbert is an accomplished scientist with extensive knowledge of robotic
s .')
```

```
Sentence:
 professor gilbert is an accomplished scientist with extensive knowledge of robotics .

Nearest neighbors:
 1. chandra is a gifted geneticist and an expert in nanotechnology . (0.240)
 2. dr. suki is a skilled scientist in the area of chemistry . (0.255)
 3. dr._tempest_bell is an intelligent scientist specializing in astrobiology . (0.262)
 4. the master is a highly proficient engineer and scientist , specializing in the use of an_alien
technology of undetermined origin . (0.272)
 5. walker is an accomplished inventor with skill in electronics and engineering . (0.277)
 6. meranno is a highly intelligent and gifted research scientist . (0.288)
 7. lord kofi_whitemane is a member of the kymellian race , who have built a peaceful , benevolent
civilization with highly advanced technology . (0.289)
 8. dr. howard is a licensed to practice medicine and is a skilled surgeon . (0.289)
 9. kaga is an extremely intelligent and wealthy individual with access to vast resources . (0.29
7)
 10. dr._tempest_bell is a young , gifted scientist and the world 's leading expert on astrobiolog
y . (0.298)
```

In [25]:
```
# more subtle example
get_sim_sents('hector told logan where to find rojas , and was killed by felix .')
```

```
Sentence:
 hector told logan where to find rojas , and was killed by felix .

Nearest neighbors:
 1. chapman encountered mister_fantastic , invisible_woman and black_panther and traveled to hong_
kong where he was killed by dolph . (0.284)
 2. by order of bastion , pierce blew up the black birds , and was killed by cyclops . (0.285)
 3. eventually , wild_child investigated nemesis 's disappearance and met up with the children_of_
the_night , and was captured by rok . (0.294)
 4. masters managed to convince bobby that the black_rider he encountered was an_impostor and left
to rescue marie . (0.314)
 5. during the attack his_sister was injured and essex logan stepped into believing he was dead an
d prompting him to kill essex . (0.317)
 6. the_vision broke bauer out of the camp and assisted him in fleeing to portugal . (0.318)
 7. clea and doctor_strange escaped dormammu , but met with umar , who wanted to kill doctor_stran
ge . (0.319)
 8. micro tried to recruit punisher but was killed by frank . (0.322)
 9. by this time , erda was captured and being held hostage by schultz . (0.328)
 10. jackson helped spider-man and thunderstrike discover the hideout of pandara in a warehouse ,
 and bring him to justice , only to sacrifice his life to save thunderstrike . (0.329)
```

In [26]:
```
# an unseen example
get_sim_sents('they were surrounded , with no escape in sight')
```

```
Sentence:
 they were surrounded , with no escape in sight

Nearest neighbors:
 1. they were pinned down by enemy fire . (0.739)
 2. they all seemed to have suffered in a battle they had no memory of . (0.751)
 3. hovering unnamed in their only appearance (0.751)
 4. each was armed with a drill on the front , allowing them to bore into anything in their path .
(0.754)
 5. the vdbn were notorious for attacking anything in sight and then fighting among themselves whe
n there were no enemies in sight . (0.757)
 6. a mob formed , attacking and destroying everything in sight , with only sheldon helping the in
jured . (0.757)
 7. there were numerous tunnels stretching out of sight , many unexplored . (0.757)
 8. there , they would save the inhabitants from the savage hairy_ones and continue on through the
valley_of_the_mists . (0.762)
 9. the x-men themselves had set a trap around to them , bringing the phoenix-powered namor and a
 squad of x-men . (0.762)
 10. together with the leatherneck_raiders , they were trapped by hydra . (0.763)
```

Other Relevant Embeddings

- GloVe
  - more explicit training method
- fastText
  - by the author of word2vec, new subword vectors, classifier

Recommended Reading

- Deep Learning, NLP, and Representations (http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/)
- Meanings are Vectors (http://sanjaymeena.io/tech/word-embeddings/)
- Vector Representations of Words (https://www.tensorflow.org/tutorials/word2vec)
- A Word is Worth a Thousand Vectors (http://multithreaded.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/#footnote1)
- On word embeddings (http://sebastianruder.com/word-embeddings-1/)