Computing Distributed Representations for Polysemous Words

Haiqing Wang

RWTH Aachen University Computer Science 5
and
Fraunhofer IAIS

Advisors:
Prof. Gerhard Lakemeyer
Prof. Christian Brauckhage, IAIS
Dr. Gerhard Paass, IAIS
Dr. Joerg Kindermann, IAIS

August 9, 2016

Overview

Word Embedding

Definition

Word embedding is a process to transform words in natural language text to vectors of real numbers.

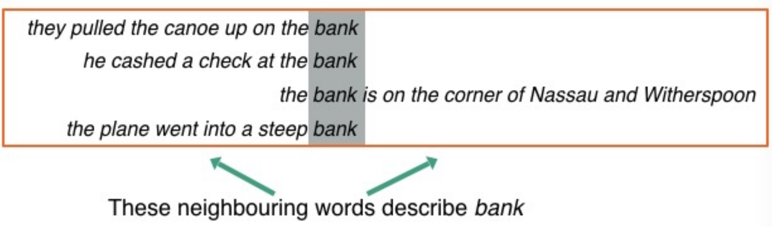
Assumption

If the contexts of two words are similar, their meaning and their vectors should be similar as well. [Harris, 1954]

Methods

LDA, LSA, Neural Network.

Word Embedding



Sense Embedding

Problem

Word Embedding approaches map each word to a single embedding vector. A word may have several different meanings. For example the word "bank" among others may designate:

- the slope beside a body of water,
- a financial institution,
- a flight maneuver of an airplane.

Target

One embedding for each sense

Methods

Most of sense embedding models are based on the word embedding models

An example result of Our Model

Cosine Similarity

$$cos(a, b) = \frac{\sum_{i=1}^{d} a_i b_i}{\sqrt{\sum_{i=1}^{d} {a_i}^2}\sqrt{\sum_{i=1}^{d} {b_i}^2}}$$

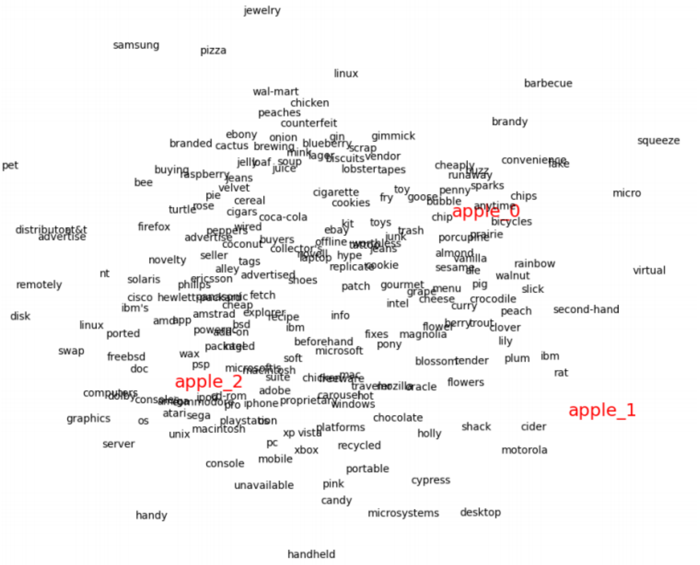
Table: Nearest words of *apple*'s different senses based on the cosine similarity

$apple_0$:	cheap , junk , scrap , advertised , gum
$apple_1$:	chocolate, chicken, cherry, berry, cream
$apple_2$:	macintosh, linux, ibm, amiga, atari

- $apple_0$ and $apple_1$ are about food, they are similar
- $apple_2$ is about the computer company

Nearest words of *apple*

Projection of 50-dimensional vectors to 2 dimensions by t-SNE



Word2Vec

Definition

- corpus $C = (S_1, S_2, \dots, S_M)$
- sentence $S_i = (w_{i,1}, w_{i,2}, \dots, w_{i,L_i})$, where L_i is the length of sentence S_i.
- $w_{i,j} \in D$: the word token from the vocabulary D in the position j of sentence S_i.
- $V_w \in \Re^d$: input embedding vector of word w
- $U_w \in \Re^d$: output embedding vector of word w

Task

predict output embeddings of neighbouring word from input embeddings. $->$ simultaneously train embeddings

Word2Vec

$Context(w_{i,t}) = (w_{i,t-c}, \ldots, w_{i,t-1}, w_{i,t+1}, \ldots, w_{i,t+c})$, and c is the number of words before and after $w_{i,t}$ in $Context(w_{i,t})$

Word2Vec ([?]) contains two different models: the CBOW model (Continuous Bag-of-Words Model) the Skip-gram model(Continuous Skip-gram model).

The CBOW model's objective function:

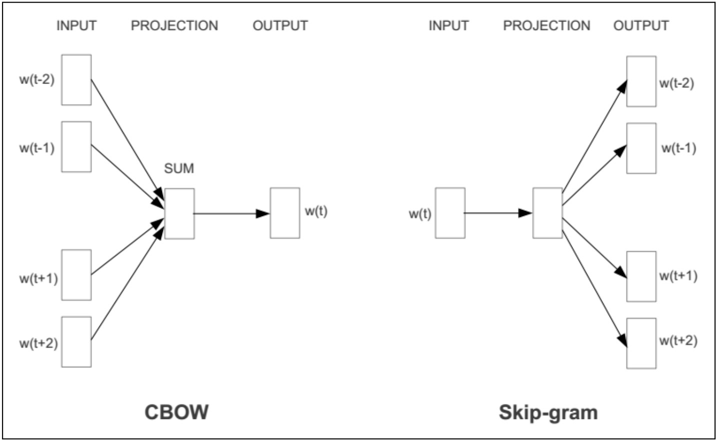
$$\prod_{i=1}^{M} \prod_{t=1}^{L_i} p(w_{i,t} | Context(w_{i,t}))$$

The Skip-gram model's objective function:

$$\prod_{i=1}^{M} \prod_{t=1}^{L_i} p(Context(w_{i,t}) | w_{i,t})$$

Word2Vec

The structures of CBOW model and Skip-gram model when $c = 2$:



Skip-Gram Model

Product all word pair prediction probabilities:

$$p(Context(w_{i,t})|w_{i,t}) = \prod_{\substack{-c \leq j \leq c \\ j \neq 0 \\ 1 \leq t+j \leq L_i}} p(w_{i,t+j}|w_{i,t})$$

$V_w \in \Re^d$: input vector of word w
$U_w \in \Re^d$: output vector of word w

Softmax Function

$$p(w_{i,t+j}|w_{i,t}) = \frac{\mathrm{e}^{U^{\mathrm{T}}_{w_{i,t+j}} V_{w_{i,t}}}}{\sum_{w \in D} \mathrm{e}^{U^{\mathrm{T}}_{w} V_{w_{i,t}}}}$$

Skip-Gram Model

Negative Sampling

- idea: replace the set of all words in denominator by a sample
- probability score of actual words should be large
- probability score of all other words should be small

$$p(w_{i,t+j}|w_{i,t}) \approx \sigma({U_{w_{i,t+j}}}^{\mathrm{T}} V_{w_{i,t}}) * \prod_{k=1}^{K} \mathbb{E}_{z_k \sim P(w)}[1 - \sigma({U_{z_k}}^{\mathrm{T}} V_{w_{i,t}})]$$

Skip-Gram Model

Negative Sampling

$$p(w_{i,t+j}|w_{i,t}) \approx \sigma({U_{w_{i,t+j}}}^{\mathrm{T}} V_{w_{i,t}}) * \prod_{k=1}^{K} \mathbb{E}_{z_k \sim P(w)}[1 - \sigma({U_{z_k}}^{\mathrm{T}} V_{w_{i,t}})]$$

where $\sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}$, K is the number of negative samples

and $P(w)$ is the smoothed unigram distribution :

$$P(w) = \frac{count(w)^{\frac{3}{4}}}{|C|^{\frac{3}{4}}}$$

(idea: emphasize the meaning bearing less frequent words)
where $|C|$ is the size of corpus, $count(w)$ is the number of times w occurred in C

Our Model

It is an extension of Skip-Gram Model.

Main Idea

- Each word in the sentence can have one or more senses.
- Each word is assigned a sense randomly and can be adjusted afterwards based on the score function.

Example

$$\text{the}_0 \text{ plain went}_0 \text{ into}_1 \text{ a}_1 \text{ steep bank}_2$$
$$- > \text{the}_0 \text{ plain went}_1 \text{ into}_1 \text{ a}_2 \text{ steep bank}_1$$

For this version we assume that the number of senses is given. We assign up to 3 senses to words with higher frequency

Our Model

Definition

- corpus $C = (S_1, S_2, \ldots, S_M)$
- sentence $S_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,L_i})$, where L_i is the length of sentence S_i.
- $w_{i,j} \in D$: the word token from the vocabulary D in the position j of sentence S_i.
- Each word $w \in D$ has $N_w \geq 1$ senses.
- $h_{i,j}$: the sense index of word $w_{i,j}$ $(1 \leq h_{i,j} \leq N_{w_{i,j}})$.
- $V_{w,s}$: input embedding vector of word w's s-th sense
- $U_{w,s}$: output embedding vector of word w's s-th sense

Our Model

Sense Prediction

- Center word's sense: $(w_{i,t}, h_{i,t})$
- neighbour word's sense: $(w_{i,t+j}, h_{i,t+j})$
- Sense prediction probability: $p((w_{i,t+j}, h_{i,t+j})|(w_{i,t}, h_{i,t}))$

The goal is to maximize the production of all sense prediction probabilities like:

$$p\Big[(w_{i,t+j}, h_{i,t+j})|(w_{i,t}, h_{i,t})\Big]$$

Our Model

Objective Function

We use a log likelihood on all possible sense prediction probabilities:

$$G = \sum_{i=1}^{M} \sum_{t=1}^{L_i} \sum_{\substack{-c \leq j \leq c \\ j \neq 0 \\ 1 \leq t+j \leq L_i}} \log p\Big[(w_{i,t+j}, h_{i,t+j}) | (w_{i,t}, h_{i,t})\Big]$$

Our Model

Negative Sampling

$$\log\ p\left[(w_{i,t+j}, h_{i,t+j})|(w_{i,t}, h_{i,t})\right] \approx \log\ \sigma(U_{w_{i,t+j},h_{i,t+j}}{}^{\mathrm{T}} V_{w_{i,t},h_{i,t}})$$

$$+\sum_{k=1}^{K} \mathbb{E}_{z_k \sim P_n(w)} \log\ [1-\sigma(U_{z_k,R(N_{z_k})}{}^{\mathrm{T}} V_{w_{i,t},h_{i,t}})]$$

where $R(x)$ is the random number between 1 and x
and $(z_1, R(N_{z_1}))$,...,$((z_K, R(N_{z_K}))$ are the negative sample words with random assigned senses

Our Model

Initialization

Initialize sense vectors with word vectors from Skip-Gram model adding a small variance; Initialize sense indexes randomly.

Assign (Update h)

Use the **score function** (sum of log probability) to select the best sense of the center word $w_{i,t}$. And it uses above process to adjust senses of whole sentence and repeats that until stable (not changed).

$$h_{i,t} = \arg\max_{1 \le s \le N_{w_{i,t}}} \sum_{\substack{-c \le j \le c \\ j \ne 0 \\ 1 \le t+j \le L_i}} \log p[(w_{i,t+j}, h_{i,t+j}) | (w_{i,t}, s)]$$

Our Model

Learn (Update V and U)

Use the gradient of the **loss function** (negative log likelihood for each sense prediction) as following to update the input embedding V and output embedding U

$$loss\big((w_{i,t}, h_{i,t}), (w_{i,t+j}, h_{i,t+j})\big)$$
$$= -\log p\Big[(w_{i,t+j}, h_{i,t+j}) \big| (w_{i,t}, h_{i,t})\Big]$$
$$\approx -\log \sigma(U_{w_{i,t+j}, h_{i,t+j}}{}^{\mathrm{T}} V_{w_{i,t}, h_{i,t}})$$
$$-\sum_{k=1}^{K} \mathbb{E}_{z_k \sim P_n(w)} \log \left[1 - \sigma(U_{z_k, R(N_{z_k})}{}^{\mathrm{T}} V_{w_{i,t}, h_{i,t}})\right]$$

Our Model

Convergence

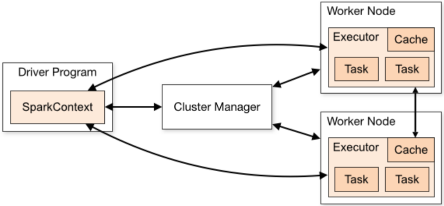
- Iterating between **Assign** and **Learn** till the convergence of the value of G makes the whole algorithm complete.
- We use the loss of validation set to monitor if the training process converges. When the validation loss begins to increase, we stop training.
- It is a stochastic equilibrium, no guarantee of an optimal state.

Implementation

We use the Apache Spark parallel execution framework to implement our model.

Spark

- Spark has one driver and several executors. Usually, an executor is a cpu core.



- *RDD* (Resilient Distributed Dataset) : a special data structure containing the items of a dataset, e.g. sentences or documents.
- Operations on *RDD*: *Transformation operations* and *Action operations*.

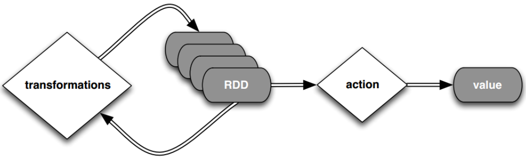
Spark

Transformation operations

A transformation operation transforms a RDD to another RDD, i.e. map (apply a function to all RDD elements) and filter (select a subset of RDD elements by some criterion)

Action operations

An action operation is used to gain some useful information from the RDD, i.e. count (count the number of items) , reduce (apply a single summation-like function) and aggregate (apply several summation-like functions)



Implementation

Data preparing

The original corpus is a snapshot of Wikipedia at April, 2010 created by , which has 990 million tokens. We remove the infrequent words. In experiments, we use two different vocabularies with size nearly 500 thousand and with size nearly 100 thousand respectively.

Training set and validation set

We split the corpus into a training set and a validation set. The training set has 99% of the data and validation set has only 1% of the data. We use the validation set to monitor our training process if it is converging.

Implementation

Embedding

- Input embedding: V; Output embedding: U
- V and U are defined as immutable broadcast variables, which are only readable and can not be changed by executors
- In each executor mutable copies of U and V are created.

Implementation

Assign Step

We use map transformation to transform each sentence with senses information to another sentence with changed senses information. V and U are used to calculate the score.

Learn Step

We transform the original sentence RDD into the two-element collection of the updated local V and updated local U in each executor and use *treeAggregate* to collect all such vectors together from different executors. And we use sum of them to be as the new V and new U.

Learning Rate Reduction

After each iteration:

$$\alpha' = \alpha * gm$$

Evaluation

Unit test of algorithm, e.g. by checking derivatives by finit difference approximation

Different Hyper-Parameters Comparison

Definition of Hyper-Parameters of the Experiments:

Fixed Parameters	
$numRDD$=20	The number of RDD to split training data set.
c=5	The size of context
K=10	The number of negative samples
Variable Parameters	
id	The id number of the experiment.
d	Vector size for each embedding vector
$c1$	Minimal count for the inclusion of a word in vocabulary D
$c2$	Count thresholds for words with two senses i.e. the count of w is more than $c2$, w has at least two senses
$c3$	Count thresholds for words with three senses i.e. the count of w is more than $c3$, w has at least three senses
lr	The learning rate at the beginning of the experiment.
gm	The reduction factor of the learning rate for each iteration
$S1$	true if sense has only one output embedding vector

Definition of Evaluation Scores:

$t1$	The average time of learning parameters in one iteration
$t2$	The average time of collecting parameters using *treeAggregate* in one iteration
$t3$	The average time of all operations in one iteration
$t4$	Total training time
$iter$	The number of total training iterations
$vLoss$	The best loss of the validation set
$SCWS$	The Spearmans rank correlations on the SCWS dataset.
$word353$	The Spearmans rank correlations on the WordSim-353 dataset

Similarity Tasks

- We perform the WordSim-353 task ([?]) and the Contextual Word Similarity (SCWS) task ([?]) to evaluate our sense embedding.
- The WordSim-353 dataset is made up by 353 pairs of words followed by similarity scores from 10 different people and an average similarity score.
- The SCWS Dataset has 2003 words pairs with their context respectively, which also contains 10 scores from 10 different people and an average similarity score.
- The two tasks are to reproduce these similarity scores and use Spearmans rank correlations ρ to calculate the correlation.

We use *avgSim* for WordSim-353 task and use both *avgSim* and *localSim* for SCWS task.

$$avgSim(w, \tilde{w}) = \frac{1}{N_w}\frac{1}{N_{\tilde{w}}}\sum_{i=1}^{N_w}\sum_{j=1}^{N_{\tilde{w}}}\cos(V_{w,i}, V_{\tilde{w},j})$$

$$localSim(w, \tilde{w}) = cos(V_{w,k}, V_{\tilde{w},\tilde{k}})$$

$$cos(a, b) = \frac{\sum_{i=1}^{d} a_i b_i}{\sqrt{\sum_{i=1}^{d} {a_i}^2}\sqrt{\sum_{i=1}^{d} {b_i}^2}}$$

where k is the most possible sense index of w given w's context and $\tilde{k}$ is the most possible sense index of $\tilde{w}$ given $\tilde{w}$'s context.

Different Number of Output Senses

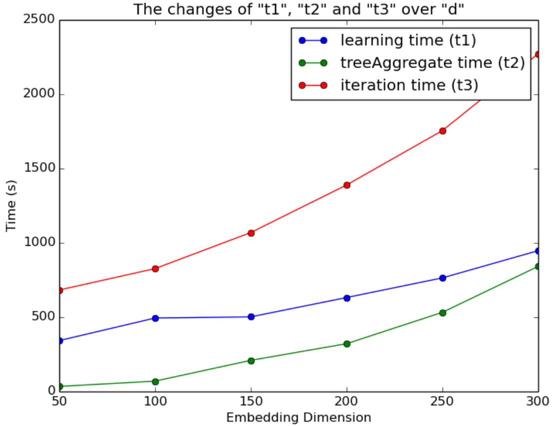
In the assumption of our model, each word has several input embeddings and several output embeddings

	id 13 , one sense output embedding	*id* 16, multiple senses output embedding
apple	cheap, junk, scrap, advertised chocolate, chicken, cherry, berry macintosh, linux, ibm, amiga	kodak, marketed, nokia, kit portable, mgm, toy, mc marketed, chip, portable, packaging
bank	corporation, banking, banking, hsbc deposit, stake, creditors, concession banks, side, edge, thames	trade, trust, venture, joint trust, corporation, trade, banking banks, border, banks, country
cell	imaging, plasma, neural, sensing lab, coffin, inadvertently, tardis cells, nucleus, membrane, tumor	dna, brain, stem, virus cells, dna, proteins, proteins dna, cells, plasma, fluid

- multiple senses output embedding has worse nearest words results

Different sizes of embedding vectors

$t1$: The average time of updating embedding
$t2$: The average time of collecting embedding
$t3$: The average iteration time



Different Min Count

$c1$: Minimal count for the inclusion of a word in vocabulary D
$t4$: Total training time
$iter$: The number of total training iterations
$vLoss$: The best loss of the validation set

id	*c1*	*t1*	*t2*	*t3*	*t4*	*iter*	*vLoss*	*SCWS*	*word353*
6	200	342.9	34.6	683.3	23915	35	0.2458	0.4666	0.4838
13	20	849.0	343	1838.1	64335	35	0.2457	0.4371	0.4293

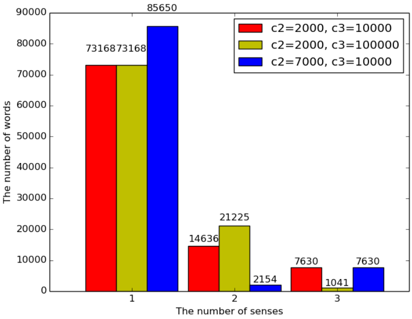
- bigger $c1$, smaller dictionary
- smaller dictionary, better similarity task scores :focuses on those more frequent words and can obtain more meaningful information

Different Sense Count Comparison

$c2$: Count thresholds for words with two senses

$c3$: Count thresholds for words with three senses

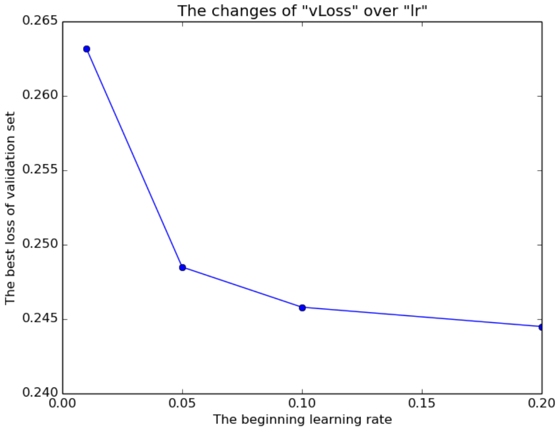
id	*c2*	*c3*	*t1*	*t2*	*t3*	*t4*	*iter*	*vLoss*	*SCWS*	*word353*
13	2000	10000	849	343	1838	64335	35	0.2457	0.4371	0.4293
14	2000	100000	798	338	1712	59912	35	0.2465	0.443	0.4375
15	7000	10000	808	340	1740	60909	35	0.2462	0.4351	0.4412



- fewer parameters, fewer time

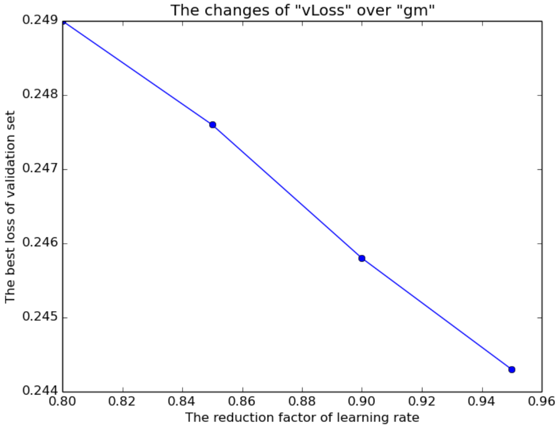
Different Learning Rate

lr : The learning rate at the beginning of the experiment.
vLoss : The best loss of the validation set



Different Gamma

gm : The reduction factor of the learning rate for each iteration
vLoss : The best loss of the validation set



Comparison to prior analyses

Table: Experimental results in the SCWS task. The numbers are Spearmans correlation $\rho \times 100$

Model	avgSim	localSim
Our Model-50d	55.8	46.7
Our Model-300d	56.9	50.5
Huang et al-50d	62.8	26.1
MSSG-50d	64.2	49.17
MSSG-300d	67.2	57.26
NP-MSSG-50d	64.0	50.27
NP-MSSG-300d	67.3	59.80

Table: Results on the WordSim-353 dataset

Model	$\rho \times 100$
Our Model-50d	48.4
Our Model-300d	48.2
C&W*	49.8
C&W	55.3
Huang et al	64.2
Skip-gram-300d	70.4

Possible reasons for bad results

- we do not use sub-sampling used word2vec
- our model do not remove the stop words
- our training is not enough and we uses too many executors (32 cores), where fewer executors may give us better results

Case Analysis

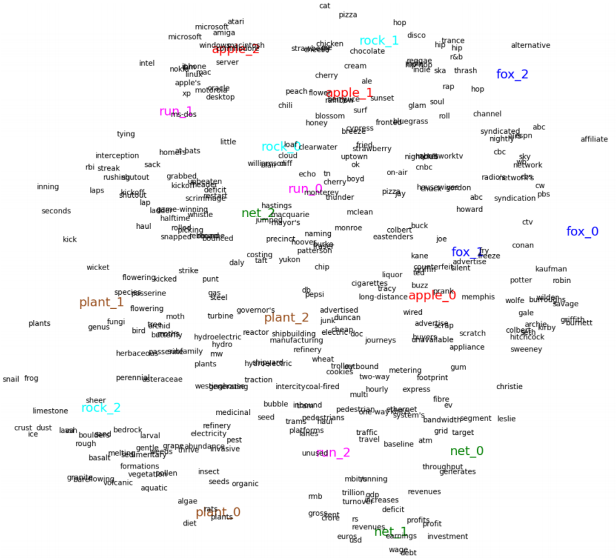
Nearest words for different senses:

fox	archie, potter, wolfe, hitchcock, conan, burnett, savage buck, housewives, colbert, eastenders, howard, kane, freeze abc, sky, syndicated, cw, network's, ctv, pbs
net	generates, atm, footprint, target, kbit/s, throughput, metering trillion, rs, earnings, turnover, gross, euros, profit jumped, rolled, rebound, ladder, deficit, snapped, whistle
rock	echo, surf, memphis, strawberry, clearwater, cliff, sunset r b, hip, roll, indie, ska, indie, hop formations, crust, melting, lava, boulders, granite, dust
run	blair, taft, fraser, monroe, precinct, mayor's, governor's streak, rushing, tying, shutout, inning, wicket, kickoff running, tram, travel, express, trams, inbound, long-distance
plant	plants, insect, seeds, seed, pollen, aquatic, organic flowering, orchid, genus, bird, species, plants, butterfly electricity, steel, refinery, refinery, manufacturing, gas, turbine

- fox: Sense 1 and 2 cover different movies and film directors while sense 3 is close to tv networks.
- net: Sense 1 is related to communication networks, sense 2 to profits and earnings and sense 3 to actions
- rock: Sense 1 and sense 2 is related to music while sense 3 to stone.
- run: Sense 1 is related to election campaigns, sense 2 expresses the movement and sense 3 to public transport.
- plant: Sense 1 is close to biologic plants and small animals, sense 2 is related to flowers and sense 3 to factories.

Case Analysis

projection of words and their neighbors to two dims by t-SNE



Conclusion

Possible reasons of suboptimal score on similarity tasks:

- we did not yet find the best combination of hyper parameters
- we had no time to optimize the configuration of Spark (e.g. memory assignment, number of data batches collected in RDDs, etc.) to be able to do an exhaustive training on many cluster nodes.

TO DO:

- try bigger sizes of embedding vectors
- modify preprocessing such as to remove the stop words
- increase the number of senses and try to extend our model so that it can decide the number of senses for each word similar to ([**?**])
- do more experiments for the different hyper-parameters in the future to make our results more reliable.

References I

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003).
A neural probabilistic language model.
journal of machine learning research, 3(Feb):1137–1155.

Collobert, R. and Weston, J. W. (2008).
A unified architecture for natural language processing: Deep neural networks with multitask learning.
In *Proceedings of the 25th international conference on Machine learning (ICML)*. ACM.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012).
Improving word representations via global context and multiple word prototypes.
In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

References II

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013).
Distributed representations of words and phrases and their compositionality.
In *Advances in neural information processing systems*, pages 3111–3119.

Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. (2015).
Efficient non-parametric estimation of multiple embeddings per word in vector space.
arXiv preprint arXiv:1504.06654.

Tian, F., Dai, H., Bian, J., Gao, B., Zhang, R., Chen, E., and Liu, T.-Y. (2014).
A probabilistic model for learning multi-prototype word embeddings.
In *COLING*, pages 151–160.

The End