[Part 1][1] - W2V model & Architecture Skipgram. Loss: reduce mean of noise-contrastive estimation training loss. Optimizer: I started using GradientDescentOptimizer, but when I used with learning rate $\simeq 1$ I had good performance up until 3k accuracy and then swings in local max, if $\simeq 0.5$ too slow to obtain any better. Inspired by simulated annealing I tried a tensor with exponential decay as learning rate (reaching 4k acc.), then with linear decay (5/6k acc). By moving to AdagradOptimizer which better handles decaying, plus early stopping on max, I was able to reach 7.69k accuracy $\simeq 52.6\%$.

[Part 1][2] - Hyperparameters (H) a. plot acc loss vs Hyperparameters 1

b. findings and final Hyperparameters Main issues: large space to search and hidden interdependencies between Hs, some intuitive like small batch size needs more iterations, high learning rate swings if too many iterations, as vocabulary grows you get more accuracy if embedding grows a bit. Other less intuitive like the smaller the batch the more you can increase negative sample. Also the behavior of Hs given few iterations is not preserved in larger number of iterations.

Final Hs: Batch 256, window 10, embedding 280, neg. sample 60, vocab. 100k, iteration 48M (34 epoch).

[Part 2][1] - Train & Visualize see vecviz.png.

[Part 2][2] - Most Similar Words In square brackets some nearby interesting words that doesn't top 5.

german germany, germans, austrian, polish, italian, [nazi, dutch, adolf, reich].
majority [1] minority, vote, support, votes, percent [confidence, share, popular, total].
general appointed, serving, representative, appointment, special [leadership, elected].
food foods, meat, nutrition, dietary, consumption [grain, products, farmers, milk].
cat cats, dog, dogs, pet, wild [animal, feline, tiger, mice].
eat eaten, eating, consume, ate, feed [drink, prey].
teach taught, teaching, learn, teaches, learned [studying, instructor, enrolled, school].

Overall they make sense, general words tend to be dirty but most of the other seems to be well captured. Seems that a good part of the understanding ability thus of the expressive power of the embeddings is wasted on inflated forms. Also mid range distance concept seems to be well abstracted (i.e., teach-enrolled, food-product, eat-prey).

[Part 3][1] - Features extraction I started using a weighted (freq. in doc over freq. in dataset) centroid and a bunch of classifiers from sklearn, but I had a very low precision ($\simeq 60\%$, with a vocabulary of 70k and embedding 128). I decided to both improve the quality of the embeddings, moving accuracy from $\simeq 40\%$ to $\simeq 52\%$, vocab. to 100k, embeddings to 280, and I re-engineered feature used to represent a document as follows.

Let be V a set of vectors and max_vec_V (respectively min_vec_V and avg_vec_V) the vector in which each dimension i is $\max_{v \in V} v[i]$ (respectively min and avg). Lets consider a document (respectively a domain) as the set of its words (respectively of the word in all the documents in the domain) by means of embedding vectors. Each document d to classify in a set DM of domains is represented by a feature vecture defined by concatenation of the following vectors:

$$[max\_vec_d, min\_vec_d, avg\_vec_d, MAXCS_d, MINCS_d, AVGCS_d]$$

Where $MAXCS_d$ (respectively $MINCS_d$ and $AVGCS_d$) is the vector:

$$[cosine\_similarity(max\_vec_d, max\_vec_{dm}) \mid \forall dm \in DM]$$

Then the TRAIN set, represented as explained, is used to train some classifiers; at runtime the one performing best (i.e., linear svm) over the DEV set is choosen to classify the TEST set docs.

[Part 3][2] - Evaluation Trained LinearSVM → F1: 0.7179, F1 weighted: 0.8384, recall: 0.8456, error percentage: 15.4369, precision weigted: 0.8443. For domain specific values see classifier_stats.txt, and for confusion matrix see conf_matrix_LSVM.csv

[1] as suggested in class I used a similar word since most/mostly are in stopwords of all my best emeddings.

Appendices

A Acuracy vs Loss

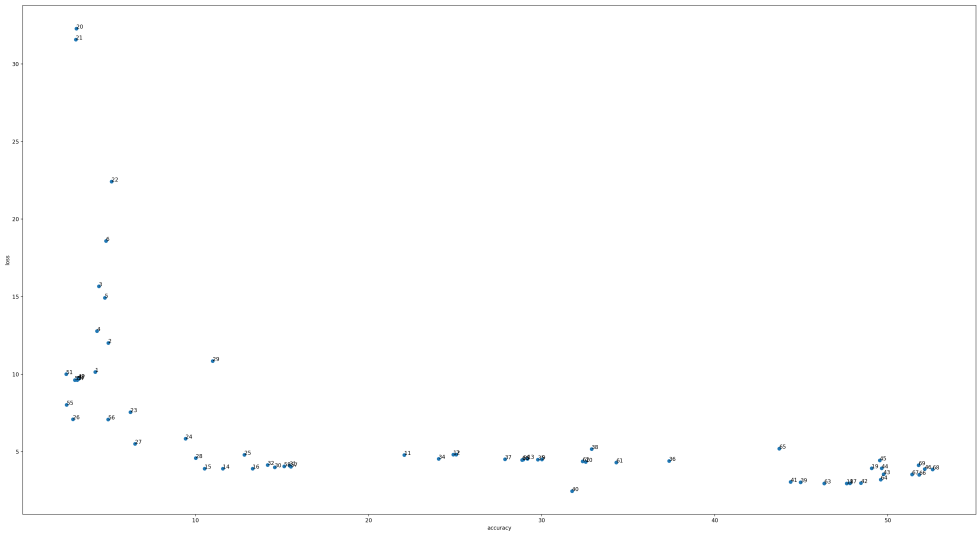

Figure 1: X: accuracy %, Y: loss, label: index to Hyperparameters (H) 1, zoom in to see details.

Table 1: Hyperparameters (H) used in 1

id	batch	embeddings	window	neg	vocab	domain	iterations	lr
1	32	200	2	200	50000	-1	6400000	1.0
2	32	200	2	200	50000	-1	12800000	1.0
3	32	200	2	200	50000	-1	25600000	1.0
4	32	200	2	200	50000	-1	12800000	1.0
5	32	200	2	200	50000	-1	6400000	1.0
6	32	200	2	200	50000	-1	12000000	1.0
7	256	128	10	20	70000	-1	24000000	1.0
8	256	128	10	20	50000	-1	24000000	1.0
9	256	280	10	60	100000	-1	24000000	1.0
10	32	200	1	200	100000	-1	1600000	1.0
11	32	200	1	200	100000	-1	3200000	1.0
12	32	200	1	200	100000	-1	3200000	0.5
13	32	300	1	300	300000	-1	3200000	1.0
14	32	200	1	200	100000	-1	6400000	0.7
15	32	200	1	200	100000	-1	6400000	1.0
16	32	200	1	200	100000	-1	6400000	1.0
17	32	200	2	200	50000	-1	12800000	1.0
18	32	200	2	200	50000	-1	12800000	1.0
19	32	200	2	200	25000	-1	12800000	1.0
20	32	200	2	200	50000	-1	12800000	1.0
21	32	128	10	200	50000	-1	24000000	1.0
22	256	280	10	20	50000	-1	12000000	1.0
23	512	128	2	20	70000	-1	12000000	1.0
24	256	280	10	20	70000	-1	12000000	1.0
25	256	280	10	20	70000	-1	24000000	1.0
26	256	280	10	40	70000	-1	24000000	1.0
27	256	280	10	60	70000	-1	24000000	1.0
28	256	280	10	100	70000	-1	24000000	1.0
29	256	280	10	60	70000	-1	48000000	1.0
30	32	200	1	200	100000	-1	6400000	1.0
31	32	200	1	200	100000	-1	6400000	1.0
32	32	200	2	200	50000	-1	12800000	1.0
33	32	200	2	200	50000	-1	12800000	1.0
34	32	200	2	200	50000	-1	25600000	1.0
35	512	128	2	200	70000	-1	12000000	1.0
36	256	128	10	20	50000	-1	24000000	1.0
37	256	128	10	20	50000	-1	24000000	1.0
38	256	280	10	200	70000	-1	24000000	1.0
39	256	280	10	40	100000	-1	48000000	1.0
40	256	128	10	40	70000	-1	48000000	1.0
41	256	280	10	60	100000	-1	48000000	1.0
42	256	280	10	80	100000	-1	48000000	1.0