Assignment 4

October 2021

Vilém Zouhar | vilem.zouhar@gmail.com

NN-1

The output class distribution is shown in Table 1. Thus, the training accuracy is 33.51% by always predicting `ORG`. In comparison to the final performance on a validation set presented in NN-5, 76.93%, this looks like a reasonable result.

Class	Rel. Frequency
CARDINAL	14.88%
DATE	7.93%
GPE	17.97%
ORG	33.51%
PERSON	25.71%

Table 1: Distribution of target classes in the provided data

NN-2

Dropout creates a filter that is applied element-wise. With probability p the original value (output of the previous layer) is replaced with 0, which prevents overfitting (regularization technique). It can be compared to bagging because of the model averaging, though in the case of dropout, this happens in the space of a single neuron. During inference, this probabilistic filter is not applied. This creates a possible issue because the input distribution is suddenly higher (e.g. previous average to one neuron was 2 and without 20% dropout is 2.5). A solution that is often implemented is scaling the input during inference by $(1 - p)$.

In my model, I added dropout after first two dense hidden layers (not at the classification layer) with 20% probability. I may have been too aggressive with dropout because the training performance was comparable to that of the validation set. Overall the validation accuracy was better than without it.

NN-3

For an overview, I set the initial number of epochs to 1000 and monitored the validation test accuracy to get some idea about what a good number of epochs could be (early stopping). Afterwards, I run cross-validation on a smaller sample of the total number

of epochs. Ideally one would use a gridsearch together with this because the learning rate is tightly coupled with how long one should train the model.

Such early stopping is usually based on an extrensic metric (accuracy on the validation set) or intrinsic one (norm of updates). I chose the former one because of its simplicity and it worked reasonably well.

Training for too long is not good because even with some regularization techniques, the network can memorize patterns in the training set and thus increase generalization error.

NN-4

Examined hyperparameters/architecture decisions:

- Optimizer: both hyperparameters had a large impact on the performance
  - Learning rate
  - Weight decay
- Number of epochs: the validation performance varied up to one percent among epochs and hence it was difficult to estimate which one is the best one
- Batch size: did not appear to have a large impact
- Architecture: was important but I focused mostly on architectures with 2 or 3 layers
  - Number of hidden layers
  - Number of neurons in the hidden layer
- Activation function: from ReLU, leaky ReLU, TanH and Sigmoid, the ReLU had the best results
- Dropout: without it, the model was overfitting
- Loss: KL-divergence and categorical cross-entropy had basically the same results
- Label smoothing: the idea was to prevent incorrectly labelled examples have a large effect on the parameter updates but it did not help
- Regularization: L1 and L2 regularization of the weights in the dense layers, no improvement

The most important parameters were related to the optimizer (type, learning rate and weight decay), the architecture (number of hidden layers, number of neurons) and the number of epochs.

My search for the best configuration could be summarized as iterative one-dimensional gridsearch. Some hyperparameters had too negligible effect on the accuracy (label smoothing, loss) and hence were omitted in future searches in step 2.

```
1: start with random (reasonable) hyperparameters
2: for every hyperparameter h:
3:   find optimum of h on the validation set
4:   set model hyperparameter to this optimum
5: if noticeable improvement in performance occurred, goto 2
```

Obviously, this algorithm is far from perfect and can easily converge to a local maximum. In the end, this was harder to evaluate because of the diminishing improvements in performance and noise in evaluation. I may have also overused the 10% of the development set and overfitted the hyperparameters by making too many decisions based on the reported results.

NN-5

I started with the baseline model and changed the optimizer to Adam with weight decay of 0.01. This single change led to the biggest improvement from validation accuracy of 64.85% to 76.79%. I then performed the aforementioned hyperparameter-tuning algorithm and converged to the following configuration:

Architecture:	L(300, 48) ∘ ReLU ∘ Dropout(0.2) ∘ L(48, 48) ∘ ReLU ∘ Dropout(0.2) ∘ L(48, 5) ∘ Softmax
Learning rate:	0.0015
Weight decay:	0.008
Bach size:	64
Epochs:	11
Loss:	Categorical cross-entropy
Label smoothing:	0
Regularization:	none

After the submission of the first model which scored 76.6% on the hidden test-set, I began using 10-fold CV with 5 runs to find out whether changing the parameters leads to improvements. I made minuscule changes (e.g. changing the number of epochs from 11 to 10 or decreasing batch size) though they did not yield better results on the test set and improved on the validation set probably due to chance. In the end, I simply trained 50 models (single training on 10 epochs took $\sim$7 seconds) and put them in a voting ensemble.

NN-6

Using the ensemble I was able to identify cases in which the voting was not overwhelmingly in favour of a single class. Such cases were almost always unclear even to me, a possible human annotator. They are shown in Table 2.

Given the training data, I think that it's not possible to increase the performance further by a large margin. Specifically, samples with numbers seem questionable when

Token	Predicted	True
37	CARDINAL	DATE
1000	CARDINAL	DATE
04	DATE	CARDINAL
deby	PERSON	ORG
woolworth	PERSON	ORG
stone	PERSON	ORG
literacy	ORG	PERSON
catholicism	ORG	GPE
medina	PERSON	ORG
mar	ORG	GPE
changzhou	GPE	ORG

Table 2: Examples of incorrectly classified samples

taken out of context and e.g. the classification of *37* as a date seems like a mistake in annotation (see the rationale for label smoothing in NN-4). I find the incorrect prediction of *woolfsworth* surprising. The surface form could serve as a last name but this is not utilized in Glove and one would expect that the word would occur in places where organizations/other retailer chains occur.

WE-1

I first examined the homonym *address* to see if it's closer to the meaning of a location or of speaking to someone. I expected the former because of the frequency. The output begins my morphological derivates of the word (including the quite infrequent verb) but then follows a list of email addresses. In fact, out of top 500 closest words (I modified the source `distance.c`), 229 are email addresses. The explanation is clear: the email address usually follows after the word *address*. Examples are listed in Table 3.

Word	Cosine Distance
addresses	0.741276
addressing	0.650099
addressed	0.648243
Address	0.641312
addresss	0.602537
adress	0.579881
addres	0.547298
Email_Brummett	0.527992
Joab_e_mail	0.517807
solve	0.503041
Joab_Jackson@idg .com	0.490110
kgray@unionleader .com	0.476449
blewis@dentonrc .com	0.467501
jrutter@lnpnews .com	0.467155

Table 3: Closest words to the input *address*

Word	Cosine Distance
as	0.614120
poorly	0.558148
nicely	0.548434
excellently	0.515061
decently	0.497770
good	0.477837
reasonably	0.468224
far	0.451972
much	0.445906

Table 4: Closest words to the input *well*

Word	Cosine Distance
surged	0.805620
climbed	0.804511
soared	0.769516
fell	0.768852
tumbled	0.725606
dipped	0.720985
jumped	0.699768
inched	0.679103
risen	0.668675

Table 5: Closest words to the input *rose*

The word *well*, also being a homonym, has the following closest words shown in Table 4. I expected the words to be of similar sentiment, though this is not what happened (e.g. poorly). Nevertheless, they occur in similar contexts. None of the words seemed to refer to the structure used to retrieve water.

Interestingly, the words related to *rose* only concern the movement and not the plant (which was more expected), as shown in Table 5 In contrast to this, the homonym *stalk* has both words related to the verb and the part of the plant in its vicinity (shown in Table 6). This was expected as neither of the two seem to be more prevalent (there is

a hypothesis that infrequent words are less likely to be retrieved). Lastly, looking at the word *first* we can notice that the similarity follows quite a good numerical ordering (expected), up to the word *ten* (unexpected). This is shown in Table 7 (non-numeric entrie were filtered out). I have no solid explanation for this only that maybe the word representation was based on only a few samples and this is the result of noise.

Word	Cosine Distance
stalks	0.680984
stalking	0.518083
sawfly_larvae	0.516364
hornworms	0.511791
tomato_hornworms	0.509612
stalked	0.506061
vine_weevil	0.493138
stalkers	0.476341
sap_sucking	0.471879
hornworm	0.469148

Table 6: Closest words to the input *stalk*

Word	Cosine Distance
second	0.797189
third	0.693208
fourth	0.673237
fifth	0.657148
sixth	0.623786
seventh	0.591549
eighth	0.555811
ninth	0.545992
eleventh	0.538080
thirteenth	0.488215
fourteenth	0.474172
twelfth	0.473737
tenth	0.460698

Table 7: Closest words to the input *first*

WE-2

In lots of Slavic languages, there is a male and a female version of a name. This allows for using the analogy {`MALE_NAME`} `- man + woman` to generate a female version of the name. This actually works with Czech word embeddings. Unfortunately not with these ones (still generates a name of popular mostly female words). In the case of *Daniel*, the name *Danielle* is unfortunately not on the list, possibly because of low-frequency occurrence

I also tried to find out who lives in the pond using the analogy {`PLACE`} `- forest + deer`. The results are shown in Table 9. It turns out it is geese and snapping turtles. They also live in the sky. Finally, it's possible (mostly) to make things better using the formula {`PROPERTY`} `- bad + good`, as presented in Table 10. *Terrible* becomes *great*, *sad* becomes *happy* and *despair* becomes *joy*.

Formula	Word	Cosine Distance
Daniel - man + woman	Christine	0.651875
	Rebecca	0.648870
	Jennifer	0.625211
	Matthew	0.621487
	Leah	0.605731
Anthony - man + woman	Angela	0.592559
	Joseph	0.583022
	Dominic	0.573644
	Rosanna	0.572420
	Catherine	0.567700
William - man + woman	Edward	0.602360
	Margaret	0.600078
	Catherine	0.595653
	Mary	0.594282
	Pamela	0.591181

Table 8: Closest words to the computation `woman - man + X`

Formula	Word	Cosine Distance
forest - deer + pond	geese	0.522082
	ponds	0.516901
	retention_pond	0.492275
	livewell	0.491924
	polliwogs	0.487612
	snapping_turtles	0.479871
	snapping_turtle	0.477647
forest - deer + sky	geese	0.469676
	antlered_buck	0.457223
	mourning_doves	0.454269
forest - deer + sea	gulls	0.488235
	goldeneyes	0.462562
	gannets	0.454313
	whales	0.454125
	pelagic_species	0.445999

Table 9: Closest words to the computation `deer - forest + X`

Formula	Word	Cosine Distance
good - bad + sad	wonderful	0.641493
	happy	0.615434
	great	0.580368
	nice	0.568397
good - bad + despair	joy	0.573050
	utter_despair	0.555167
	hopelessness	0.548456
good - bad + terrible	great	0.765869
	terrific	0.695784
	horrible	0.674380
	fantastic	0.660054
	wonderful	0.648352

Table 10: Closest words to the computation `good - bad + X`