CEFR sentence classification

We will try to build a classifier for the sentences labeled by CEFR level of difficulty.

In[1187]:=
```
levelNames = {"a1", "a2", "b1", "b2"};
```

Reading the data

In[1188]:=
```
rootPath = NotebookDirectory[];
```

In[1189]:=
```
texts = Table[Import[ FileNameJoin[{rootPath , "data", "sentences" ,
       StringJoin[ level, " sentences.txt"]}]], {level, levelNames}];
```

In[1190]:=
```
TextSentences[texts[[1]], 4] (*Show 4 sentences, only A1 level*)
```

Out[1190]=
```
{I would like that, my dear., I would like to order  la carte.,
 I would like to compose a message., Beautiful work.}
```

In[1191]:=
```
Flatten[TextSentences[#, 1] &@texts]
(*Show 1 sentence, all levels in one array*)
```

Out[1191]=
```
{I would like that, my dear., That's Speedy!,
 And guess what?, He couldn't even guess.}
```

Pre-Processing

In[1192]:=
```
sentencesFull = Map[Select[StringSplit[#, "\n"], StringLength[#] >= 2 &] &, texts];
```

In[1193]:=
```
nSents = All;(*reduce number of sentences if too slow*)
```

In[1194]:=
```
sentences = Map[Take[#, nSents] &, sentencesFull];
```

In[1195]:=
```
nameLength = {levelNames, Map[Length, sentences]}^T;
nameLength // TableForm
```

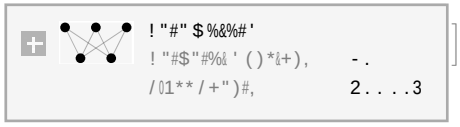
Out[1195]//TableForm=

a1	2484
a2	5110
b1	6365
b2	4799

In[1196]:=
```
labels = Flatten[Table[Table[x[[1]], x[[2]]], {x, nameLength}]];
```

Modeling

In[1225]:=
```
embedding = NetModel[
  "GloVe 50-Dimensional Word Vectors Trained on Wikipedia and Gigaword-5 Data"]
```

Out[1225]=
```
EmbeddingLayer[  ]
```

In[1226]:=
```
decoder = NetDecoder[{"Class", levelNames}]
```

Out[1226]=
```
NetDecoder[ 45$(,  /01**
            617(0*,  {138 198 738 79}
            %&'()*&+)*,  2 ]
```

In[1227]:=
```
encoder = NetEncoder[decoder];
```

In[1228]:=
```
encodedLabels = Thread[UnitVector[Length[levelNames], encoder[labels]]];
Dimensions[encodedLabels]
RandomSample[{labels, encodedLabels}^T, 4] // TraditionalForm
```

Out[1229]=
```
{18758, 4}
```

Out[1230]//TraditionalForm=

$$\begin{pmatrix} \text{b1} & \{0,0,1,0\} \\ \text{a2} & \{0,1,0,0\} \\ \text{b2} & \{0,0,0,1\} \\ \text{b1} & \{0,0,1,0\} \end{pmatrix}$$

In[1231]:=
```
encodedWords = embedding[Flatten[sentences]];
```

In[1232]:=
```
encodedSentences = Map[Mean, encodedWords];
Dimensions[encodedSentences]
```

Out[1233]=
```
{18758, 50}
```

In[1234]:=
```
allData = Thread[encodedSentences → encodedLabels]; allData[[1]]
```

Out[1234]=
```
{0.214019, 0.271261, 0.0504771, -0.276674, 0.502119, 0.0583472,
  -0.400325, -0.0207804, -0.278395, 0.0111262, -0.100528, 0.618868,
  -0.417139, -0.137525, 0.651601, 0.404062, -0.005945, 0.105084,
  0.130525, -0.439414, -0.228674, 0.428292, 0.379462, 0.0416915, 0.528543,
  -1.73525, -0.789228, 0.155749, 0.446609, -0.553966, 2.80366, 0.391172,
  -0.386574, -0.00050975, -0.116356, -0.154927, 0.13208, 0.0334483,
  0.321645, -0.272565, -0.109711, 0.119482, -0.138774, 0.242132, 0.157917,
  0.140987, -0.210705, -0.206912, -0.0672505, 0.348317} → {1, 0, 0, 0}
```

In[1235]:=
```
trainingData = RandomSample[allData, Round[0.7 Length[allData]]];
```

In[1236]:=
```
testingData = Complement[allData, trainingData];
```

In[1237]:=
```
net = NetChain[{LinearLayer[20], ElementwiseLayer[Ramp], LinearLayer[],
    ElementwiseLayer[Ramp]}, "Input" → 50, "Output" → decoder];
```

In[1238]:=
```
net = NetTrain[net, trainingData, Method → "ADAM", ValidationSet → testingData]
```

Out[1238]=
```
NetChain[ :)$"#        ;(<#+=(*i>(, - .)
          3 6i)(1=615(=  ;(<#+=(*i>(, 9.)
          9 ?1'$         ;(<#+=(*i>(, 9.)
          @ 6i)(1=615(=  ;(<#+=(*i>(, 2)
          2 ?1'$         ;(<#+=(*i>(, 2)
            !"#$"#       <01** ]
```

Evaluation

In[1239]:=
```
predict = net[Keys[testingData]];
actual = decoder[Values[testingData]];
```

In[1241]:=
```
countsNN = Counts[Table[Apply[Equal, z], {z, {predict, actual}^T}]]
```

Out[1241]=
```
⟨|True → 4232, False → 1390|⟩
```

In[1242]:=
```
countsNN[True] / Total[countsNN] // N
```

Out[1242]=
```
0.752757
```

And compare to plain Markov chain type classifier with same GloVe embeddings

In[1243]:=
```
classifier = Classify[Keys[trainingData] → decoder[Values[trainingData]]];
```

In[1244]:=
```
cm = ClassifierMeasurements[classifier,
  Keys[testingData] → decoder[Values[testingData]], "Accuracy"]
```

Out[1244]=
```
0.699217
```