Winter Semester 2009/10
Prof. Dr. Detmar Meurers
Holger Wunsch
Aleksandar Dimitrov

ISCL, Universität Tübingen
Computational Approaches
to Functional Elements

Lab Session January 13, 2010:
The Tilburg Memory-Based Learner (TiMBL)

1 First steps with TiMBL

The purpose of this task is to get a feeling of how TiMBL works, what is going on when computing distances, what happens with ties, and so on. Essentially, we explore TiMBL's behaviour in assigning names to ranges.

We start with a *very simple* training set, consisting of the following three samples:

```
0,a
1,b
5,c
```

Please save this training set in a file named `first.train`.

We also need a corresponding test set, `first.test`:

```
0,a
2,b
5,c
```

Now, before you actually run TiMBL, play TiMBL! Fill in this table with what you think TiMBL will output:

		You	**TiMBL**
0	a		
2	b		
5	c		

Now run TiMBL:

```
Timbl -f first.train -t first.test
```

Compare TiMBL's output with yours. You can find the output in the file `first.test.IB1.O.gr.k1.out`.

Are TiMBL's classifications as you expected?

Move the last line in the training set up first. Should this make a difference? If yes, why? If no, why not? What does TiMBL do, and why?

Add this additional instance to the test set:

```
2.9,b
```

Again, predict the outcome and compare the actual output to your predictions. Does the output match your intuition?

Force TiMBL to interpret the first feature as numeric by adding an option:

```
Timbl -mO:N1 -f first.train -t first.test
```

The output is now in `first.test.IB1.O:N1.gr.k1.out`. Compare this to the output using the default settings.

2 Classification by example

In this task we will look at how TiMBL uses stored examples to classify new examples.

We will teach TiMBL to classify vehicles based on this small data set:

	Horsepower	**Wheels**	**Fuel**	**Class**
VW Passat 1.6	102	4	Premium	Car
Mercedes C180	156	4	Premium	Car
VW Passat 1.6 TDI	170	4	Diesel	Car
Bike	0.14	2	Food	Bike
Caterpillar 345C L	345	0	Diesel	Excavator
Kick Scooter	0.14	2	Food	Kick-Scooter
BMW F800GS Motorcycle	90	2	Premium	Motorcycle
Honda CB 1000R	125	2	Regular	Motorcycle
Vespa GTS 250 i.e.	22	2	Regular	Scooter
Honda CN 250 Helix	17	2	Regular	Scooter
John Deere 8345R	345	4	Diesel	Tractor

Create a TiMBL test set named `vehicles.test` including all instances above. In the training set, include at first *only one* feature vector corresponding to the first car in the list. The first column is not important; omit it when translating the table to TiMBL features.

Now run TiMBL. What happens? Why?

Next gradually add additional training instances. Check out how the classifications change with the additional bits of information that TiMBL gets.

Play around with the training and test sets. Can you force ties? Can you construct training sets such that the feature ordering changes?

Note: In this toy example, we will not use separate development and test sets, and gradually, more and more instances from the test set will be added to the training set. This is okay here, as the purpose of this task is to get a good understanding of how TiMBL classification works. In a real task however, it is important to keep the three sets strictly disjoint, to ascertain as much as possible that the results properly generalize to unknown new data – which is, after all, the reason for constructing a machine learner.

3 Inflection of German articles

Let's turn to linguistics again. We will teach TiMBL how to inflect German definite and indefinite articles. More specifically, given a morphological specification such as `asm` *(accusative singular masculine)* and the article type such as `indef`, TiMBL should output *einen*.

Solve the task in these steps:

1. Extract training and test data from TüBa-D/Z 5.0 and convert these in TiMBL feature vectors
2. Run TiMBL on the data
3. Examine the results

Extraction of training and test data from TüBa-D/Z 5.0

In order to teach TiMBL to inflect articles, we need training data that describes the relevant rules, i.e. the form of a definite or indefinite article given a specific inflection. For this experiment, we will use the TüBa-D/Z treebank (verison 5) as our data source, because it contains both parts-of-speech and morphological analyses for all tokens. From the treebank, extract all articles and their morphological analyses. Note that sometimes, the morphological analysis might be underspecified.

The treebank is in the Negra-Export format (see the task on language models of articles and prepositions for German for an explanation of this format at http://purl.org/dm/09/ws/func/ex4.pdf). You can find the treebank here:

`/afs/sfs/lehre/dm/ws-09-10-functional-elements/corpora/tuebadz-5.0.export`

Include in your program some code that determines whether an extracted article is definite or indefinite.

The resulting feature vectors should look like this:

```
die.nsf,def,n,s,f,die
die.np*,def,n,p,*,die
eine.asf,indef,a,s,f,eine
```

The first feature is for reference purposes only. To exclude it from TiMBL, we use the switch `-mO:I1`. This is a practical trick if you want to include some reference data in the feature sets for later processing or inspecting.

Create a training set which contains 90% of all instances, and a test set which contains the remaining 10%.

Note: There are a number of articles with alternative spelling (abbreviated, colloquial, mistakes), which you can ignore. This is an exhaustive list:

'm, 'n, 'ne, 's, a, an, anner, ie, inen, n, ne, vom.

Running TiMBL

Assuming that you named the training set `inflect.train` and the test set `inflect.test`, run TiMBL using

```
Timbl -mO:I1 -f inflect.train -t inflect.test
```

Evaluating the results

Examine and interpret the results.

Experiment with different command line switches: Increase the number k of neighbors considered by TiMBL. Change the distance metric used. The TiMBL manual[1] lists all command line switches understood by TiMBL in section 6.

4 Reflecting on the data

As we saw in task 2 above, and also in the the language modeling tasks, machine learning is very much about the data. The question of *what* data, i.e. what features, is the most important – given bad features, the classifier will also produce bad results, garbage in, garbage out.

It also looks like that *more* data is always better than less data. Is this really true? On the basis of our example, how can this question be researched empirically?

[1] http://ilk.uvt.nl/downloads/pub/papers/Timbl_6.2_Manual.pdf