DATASETS AND READERS

Datasets

We use a number of datasets in this book. Some of them are freely available, some are freely downloadable but have licences or terms and conditions that restrict what you can do with them, some you have to pay for. We provide tools for unpacking most of the freely available datasets used in the book and putting them where they need to be, and for a few we provide tools for downloading them as well. Some of them have restrictions on what you can do with them, and most of them say that while we can write programs that do things using them we cannot directly redistribute them (but we can tell you where to download them from). You should always look at the licences before downloading them. We have tried to stick to corpora without restrictive licences where we can, and will always point to the licence conditions when we tell you where to get them from, **but it is up to you to obey the terms and conditions.**

The programs we will be using expect the corpora to be inside a directory called CORPORA. You can choose where that should be kept, but you should specify it inside the program file **basics/corpora.py**. I keep everything inside a folder called **/Library/WebServer/CGI-Executables/SENTIMENTS**, which is perhaps a slightly odd place to keep it but it works for me, so my basics/corpora.py contains the line

```
CORPORA = "/Library/WebServer/CGI-Executables/SENTIMENTS/CORPORA"
```

You need to set this to wherever you're going to want to keep the corpora.

In order to put corpora in the appropriate place inside CORPORA, they need to be downloaded. Some of the sources we use allow people to download the data and then make use of programs that manipulate it in various ways, but does not allow us to download and distribute it directly. We therefore have a directory inside CORPORA called DOWNLOADS which is where you should put the downloaded data. Our scripts will then unpack it and convert it into the form we need it to be and put it it in the correct place within CORPORA, but you will have to do the actual downloads. We will specify where to get it from and where you should put it in the relevant scripts. The general structure of this directory is as shown below (files that you will have to download are highlighted in red):

```
--|CORPORA--|DOWNLOADS--|TEXTS--|BNC--|2554.zip
                        |
                        |TREEBANKS--|UDT--|ud-treebanks-v2.12.tgz
                        |
                        |TWEETS--|CARER--|EN--|data.jsonl.gz
                                             |
                                             |dataset_infos.json
                                  |
                                  |IMDB--|EN--|aclImdb_v1.tar.gz
                                  |
                                  |SEM11--|AR--|E-c-Ar-dev.zip
                                               |
                                               |E-c-Ar-train.zip
                                          |
                                          |EN--|E-c-En-dev.zip
```

```
                                        |
                                        |E-c-En-train.zip
                                 |
                                 |ES--|E-c-Es-dev.zip
                                        |
                                        |E-c-Es-train.zip
                           |
                           |SEM4--|AR--|EI-oc-Ar-dev.zip
                                        |
                                        |EI-oc-Ar-train.zip
                                 |
                                 |EN--|EI-oc-En-dev.zip
                                        |
                                        |EI-oc-En-train.zip
                                 |
                                 |ES--|EI-oc-Es-dev.zip
                                        |
                                        |EI-oc-Es-train.zip
                           |
                           |WASSA--|EN--|anger-ratings-0to1.dev.target.txt
                                        |
                                        |anger-ratings-0to1.train.txt
                                        |
                                        |fear-ratings-0to1.dev.target.txt
                                        |
                                        |fear-ratings-0to1.train.txt
                                        |
                                        |joy-ratings-0to1.dev.target.txt
                                        |
                                        |joy-ratings-0to1.train.txt
                                        |
                                        |sadness-ratings-0to1.dev.target.txt
                                        |
                                        |sadness-ratings-0to1.train.txt
|
|TEXTS--|BNC--|download--|Doc--|HTML--|BNCdes.html
                                           |
                                           |bibliog.html
                                           ...
                                     |
                                     |Src
                                |
                                |Texts--|A--|A0--|A00.xml
```

```
                                                |
                                                |A01.xml
                                                ...
                                            |
                                            |A1--|A10.xml
                                                 |
                                                 |A11.xml
                                                 ...
                                            ...
                                    |
                                    |B--|B0--|B01.xml
                                                 |
                                                 |B02.xml
                                                 ...
                                            |
                                            |B1--|B10.xml
                                                 |
                                                 |B11.xml
                                                 ...
                                            ...
                                    ...
            |
    |
    |TREEBANKS--|UDT--|ud-treebanks-v2.12--|UD_Abaza-ATB--|LICENSE.txt
                                                         |
                                                         |wholething
                                          |
                                          |UD_Afrikaans-AfriBooms--|LICENSE.txt
                                                                   |
                                                                   |wholething
                                          ...
    |
    |TWEETS--|CARER--|EN--|wholething.csv
             |
             |IMDB--|EN--|wholething.csv
             |
             |KWT--|KWT.M--|AR--|wholething.csv
                   |
                   |KWT.U--|AR--|wholething.csv
             |
             |QATAR--|AR--|wholething.csv
             |
             |SEM11--|AR--|wholething.csv
```

```
                        |
                        |EN--|wholething.csv
                        |
                        |ES--|wholething.csv
                |
                |SEM4--|AR--|wholething.csv
                       |
                       |EN--|wholething.csv
                       |
                       |ES--|wholething.csv
                |
                |WASSA--|EN--|wholething.csv
```

Texts and Treebanks

We will be using two sorts of datasets. We use various textual resources – some just plain texts, some annotated texts, some treebanks – for developing and testing preprocessing steps like morphological analyis, identification of of compound terms and so on; and we use collections of annotated tweets for the main business of the book, i.e. developing and testing emotion identification algorithms. We start by looking at how to obtain and unpack the main textual resources.

British National Corpus (BNC)

To download and install the BNC, do the following.

Read the licence at http://www.natcorp.ox.ac.uk/docs/licence.html: you must obey the terms and conditions of this licence if you are going to use this resource. Then download it from

https://ota.bodleian.ox.ac.uk/repository/xmlui/bitstream/handle/20.500.12024/2554/2554.zip?sequence=3&isAllowed=y

and put the file **2554.zip** that you get from there in **<CORPORA>/DOWNLOADS/TEXTS/BNC** (i.e. inside a directory called **DOWNLOADS** inside the one where you're putting the corpora, which you should set in basics/corpora.py as described above). Then, in the main directory where you are keeping the program code do:

```
>>> from basics import datasets
>>> bnc = datasets.BNC()
```

If there is a file called **2554.zip** inside **<CORPORA>/DOWNLOADS/TEXTS/BNC** this will unzip it and put it in a directory with the structure shown above inside **<CORPORA>/TEXTS/BNC**, where the actual data is in files inside **A0, A1,** ... Once you've done this you can delete **2554.zip** to save space – it's quite big so you may not want to keep it lying around once you've unpacked it.

If you call **datasets.BNC()** without having downloaded **2554.zip** it will remind you of where to get it and where to put it.

```
>>> datasets.BNC()
```

```
You need to download the file 2554.zip from
https://ota.bodleian.ox.ac.uk/repository/xmlui/bitstream/handle/20.500.12
024/2554/2554.zip?sequence=3&isAllowed=y to <CORPORA>/DOWNLOADS/TEXTS/BNC
before you can install the BNC. Please check the licence details at
http://www.natcorp.ox.ac.uk/docs/licence.html before downloading it
```

Universal Dependency Treebank (UDT)

Again you should download the source for the UDT manually from

https://lindat.mff.cuni.cz/repository/xmlui/bitstream/handle/11234/1-5150/ud-treebanks-v2.12.tgz?sequence=1&isAllowed=y.

to

<CORPORA>/DOWNLOADS/TREEBANKS/UDT/ud-treebanks-v2.12.tgz

and then once you've done that you can unpack it and put it where the programs expect it to be by doing

```
>>> udt = datasets.UDT()
```

```
UNZIPPING THE MAIN SOURCE FILE
CONVERTING INDIVIDUAL TREEBANKS
REMOVING NON-CCAS FILES
```

This will put the UDT in a directory with the following structure inside **CORPORA/TREEBANKS** (it's inside **TREEBANKS** rather than **TEXTS** because the data is annotated so that it can be interpreted as a set of trees), where the actual data is in files in the wholething files inside the per language subdirectories. There is a lot more material in the UDT itself, but since we do not use it for anything we collect all the trees from files within a subdirectory into a single file called wholething and then remove everything else (except for the licence details for the subdirectory) to save space.

However, since some of the files have restrictive licences we read through all the licences and delete all the datasets which do not have Creative-Commons-Share-Alike (CCSA) licenses. CCSA licenses allow you to do most things that you would want to do (though you should check at https://creativecommons.org/licenses/ to be sure), so we keep those and delete the others.

```
--|UDT--|ud-treebanks-v2.12--|UD_Abaza-ATB--|LICENSE.txt
                              |              |
                              |              |wholething
                              |
                              |UD_Afrikaans-AfriBooms--|LICENSE.txt
                                                       |
                                                       |wholething
                              |
                              |UD_Akkadian-PISANDUB--|LICENSE.txt
                                                     |
                                                     |wholething
                              |
                              |...
```

As with the BNC, if you try to install the UDT without having downloaded the required file you'll get a message telling you where to get it from.

```
>>> udt = datasets.UDT()
```

```
UNZIPPING THE MAIN SOURCE FILE
```

```
Before you can install the UDT, you have to download the file

https://lindat.mff.cuni.cz/repository/xmlui/bitstream/handle/11234/1-
5150/ud-treebanks-v2.12.tgz?sequence=1&isAllowed=y

to

<CORPORA>/DOWNLOADS/TREEBANKS/UDT/ud-treebanks-v2.12.tgz

(<CORPORA> is the place where you have chosen to store the datasets: you
should set it in corpora.py).

The UDT consists of a collection of treebanks supplied by different peo-
ple, with a variety of licences. Different treebanks from within the
overall collection have different licences. When you download the
main .tgz file you get all of them. This downloader will remove all ex-
cept the ones with Creative-Commons-Attribution-ShareAlike licences (see
https://creativecommons.org/licenses/? for moredetails) and the .tgz file
itself when it finishes.

DO NOT USE THESE TREEBANKS UNLESS YOU AGREE TO THE TERMS OF C-C-A-S LI-
CENCES.
```

Tweets

We also need datasets containing tweets annotated with emotion labels for developing and evaluating the actual emotion classification algorithms. Again these have a range of licences, so in some cases you need to download the data yourself before using our scripts to unpack and reorganise and in others the downloader itself downloads the data as well as unpacking it. **If you want to use one the downloadable datasets, you must read the licence/terms-and-conditions and you must abide by these.**

We keep the tweet data inside CORPORA/TWEETS in directories which are in turn split into language specific directories each of which contains a single file called wholething.csv. Wholething.csv will be a tab-separated file of the following form, i.e. the first two columns are the tweet ID and the tweet itself and the remainder specify what emotions it expresses – 1 denotes that this tweet does express this emotion and 0 that it does not. This format makes it easy to allow a single tweet to denote 0 emotions (if all the emotion columns are 0) or one emotion (if exactly one emotion column is 1) or more than one (if more than one column is 1).

ID	tweet	anger	fear	joy	sadness
40000	Depression sucks! #depression	0	0	0	1
40001	Feeling worthless as always #depression	0	0	0	1

WASSA

The WASSA data comes from .https://saifmohammad.com/WebPages/EmotionIntensity-SharedTask.html The data itself is stored as a collection of files with names which indicate the emotion expressed by the tweets they contain, e.g. anger-ratings-0to1.train.txt and anger-ratings-0to1.dev.target.txt both contain tweets that express anger, and joy-ratings-0to1.train.txt and joy-ratings-0to1.dev.target.txt contain tweets that express joy. The WASSA data is split into training and development sets, but we merge these into a single file called wholething.csv because we want to be in control of which data is used for training, which is used for development and which for testing.

As usual, you have to download the datasets from the original website before you can start. In this case you need to download a set of files called

http://saifmohammad.com/WebDocs/EmoInt%20Train%20Data/anger-ratings-0to1.train.txt

http://saifmohammad.com/WebDocs/EmoInt%20Dev%20Data/anger-ratings-0to1.dev.target.txt

http://saifmohammad.com/WebDocs/EmoInt%20Train%20Data/fear-ratings-0to1.train.txt

http://saifmohammad.com/WebDocs/EmoInt%20Dev%20Data/fear-ratings-0to1.dev.target.txt

http://saifmohammad.com/WebDocs/EmoInt%20Train%20Data/joy-ratings-0to1.train.txt

http://saifmohammad.com/WebDocs/EmoInt%20Dev%20Data/joy-ratings-0to1.dev.target.txt

http://saifmohammad.com/WebDocs/EmoInt%20Train%20Data/sadness-ratings-0to1.train.txt

http://saifmohammad.com/WebDocs/EmoInt%20Dev%20Data/sadness-ratings-0to1.dev.target.txt

to

```
<CORPORA>/DOWNLOADS/TWEETS/WASSA/EN/anger-ratings-0to1.train.txt
<CORPORA>/DOWNLOADS/TWEETS/WASSA/EN/anger-ratings-0to1.dev.target.txt
<CORPORA>/DOWNLOADS/TWEETS/WASSA/EN/fear-ratings-0to1.train.txt
<CORPORA>/DOWNLOADS/TWEETS/WASSA/EN/fear-ratings-0to1.dev.target.txt
<CORPORA>/DOWNLOADS/TWEETS/WASSA/EN/joy-ratings-0to1.train.txt
<CORPORA>/DOWNLOADS/TWEETS/WASSA/EN/joy-ratings-0to1.dev.target.txt
<CORPORA>/DOWNLOADS/TWEETS/WASSA/EN/sadness-ratings-0to1.train.txt
<CORPORA>/DOWNLOADS/TWEETS/WASSA/EN/sadness-ratings-0to1.dev.target.txt
```

on your machine (remember, **<CORPORA>** is where you have set **CORPORA** to be in **basics/corpora.py**).

Once you've downloaded these files then

```
>>> wassa = datasets.WASSA()
```

will create **<CORPORA>/TWEETS/WASSA/EN/wholething.csv**, which is what we need.

SEM4/SEM11

To get the SEM4/SEM11 datasets, you have to download the source files. For SEM4 these are

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/EI-oc/English/2018-EI-oc-En-dev.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/EI-oc/English/EI-oc-En-train.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/EI-oc/Arabic/2018-EI-oc-Ar-dev.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/EI-oc/Arabic/2018-EI-oc-Ar-train.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/EI-oc/Spanish/2018-EI-oc-Es-dev.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/EI-oc/Spanish/2018-EI-oc-Es-train.zip

which you have to download to

```
<CORPORA>/DOWNLOADS/TWEETS/SEM4/EN/2018-EI-oc-En-dev.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM4/EN/EI-oc-En-train.zip
```

```
<CORPORA>/DOWNLOADS/TWEETS/SEM4/AR/2018-EI-oc-Ar-dev.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM4/AR/2018-EI-oc-Ar-train.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM4/ES/2018-EI-oc-Es-dev.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM4/ES/2018-EI-oc-Es-train.zip
```

and for SEM11 they are

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/E-c/English/2018-E-c-En-dev.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/E-c/English/2018-E-c-En-train.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/E-c/Arabic/2018-E-c-Ar-dev.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/E-c/Arabic/2018-E-c-Ar-train.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/E-c/Spanish/2018-E-c-Es-dev.zip

http://saifmohammad.com/WebDocs/AIT-2018/AIT2018-DATA/E-c/Spanish/2018-E-c-Es-train.zip

which have to be downloaded to

```
<CORPORA>/DOWNLOADS/TWEETS/SEM11/EN/2018-E-c-En-dev.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM11/EN/2018-E-c-En-train.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM11/AR/2018-E-c-Ar-dev.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM11/AR/2018-E-c-Ar-train.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM11/ES/2018-E-c-Es-dev.zip
<CORPORA>/DOWNLOADS/TWEETS/SEM11/ES/2018-E-c-Es-train.zip
```

Note the name of the English training set for SEM4, which does not fit the pattern of the names for the other SEM4 files.

Once you have these files, doing

```
>>> sem4 = datasets.SEM4()
```

```
Converting data in /Library/WebServer/CGI-Executables/SENTIMENTS/
CORPORA/TWEETS/SEM4
```

```
>>> sem11 = datasets.SEM11()
```

```
Converting data in /Library/WebServer/CGI-Executables/SENTIMENTS/
CORPORA/DOWNLOADS/TWEETS/SEM11
```

will make

```
--|TWEETS--|SEM11--|AR--|wholething.csv
                  |
                  |EN--|wholething.csv
                  |
                  |ES--|wholething.csv
           |
           |SEM4--|AR--|wholething.csv
                  |
```

```
                    |EN--|wholething.csv
                    |
                    |ES--|wholething.csv
```

As always, if you haven't download the files to the right place you will get a message telling you where to get them and where to put them.

CARER

To get the CARER dataset, download

https://huggingface.co/datasets/dair-ai/emotion/resolve/main/data/dataset_infos.json

to

```
<CORPORA>/DOWNLOADS/TWEETS/CARER/EN/dataset_infos.json
```

and

https://huggingface.co/datasets/dair-ai/emotion/resolve/main/data/data.jsonl.gz

to

```
<CORPORA>/DOWNLOADS/TWEETS/CARER/EN/data.jsonl.gz
```

Doing

```
carer = datasets.CARER()
```

will then create the required **<CORPORA>/TWEETS/CARER/EN/wholething.csv**

IMDB

To get the IMDB dataset, download

https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

to

```
<CORPORA>/DOWNLOADS/TWEETS/IMDB/EN/aclImdb_v1.tar.gz
```

(this one is quite big and will take a few minutes to download) and do

```
>>> imdb = datasets.IMDB()
UNZIPPING MAIN FILE
CONVERTING DATA FILES
```

KWT

The KWT.M and KWT.U datasets are our own and hence we can distribute them with the code, so there is nothing to download. Just do

```
>>> kwtm = datasets.KWTM()
>>> kwtu = datasets.KWTU()
```

READERS

Once we have our datasets, we have to be able to do something with them. For many tasks, we want to extract the words or the words with their tags. For this, we use the notion of readers from Chapter 4. We specify where we want to read from, what we want to use when we are reading a leaf file, and optionally a pattern that leaf files should match (we may need to do this if there are files other than actual data files stored in the directory where we are looking).

We start by thinking about the BNC. Leaf files in the BNC contain tagged sentences like the following:

```
<s n="1"><w c5="NN1" hw="factsheet" pos="SUBST">FACTSHEET </w><w c5="DTQ"
hw="what" pos="PRON">WHAT </w><w c5="VBZ" hw="be" pos="VERB">IS </w><w
c5="NN1" hw="aids" pos="SUBST">AIDS</w><c c5="PUN">?</c></s></head><p>
```

In order to extract words, with or without tags, from this we use the following regular expression:

```
<(?P<tagtype>w|c) .*?c5="(?P<c5>[A-Z0-9]*?)(-[A-Z0-9]*)?".*?>(?P<form>\S*?)\s*</(?P=tagtype)>
```

This will match an opening angle bracket followed by either w (for words) or c (for punctuation marks) followed by some stuff and then an optional tag (c5) and a closing bracket, then some non-whitespace, which is the word itself, followed by a closing </w> or </c> depending on whether the opening was w or c. There is a slight complication here because the BNC sometimes contains ambiguous tags, e.g. at one point the word care is given as <w c5="NN1-VVB" hw="care" pos="SUBST">care</w>, which says that it is not known whether this word is a noun or a verb. In such cases we assume that the first tag is correct.

We can look at all the occurrences of this pattern in the sentence above:

```
>>> s = """<s n="1"><w c5="NN1" hw="factsheet" pos="SUBST">FACTSHEET </
w><w c5="DTQ" hw="what" pos="PRON">WHAT </w><w c5="VBZ" hw="be"
pos="VERB">IS </w><w c5="NN1" hw="aids" pos="SUBST">AIDS</w><c c5="PUN">?
</c></s></head><p>"""

>>> for i in readers.BNCWordPattern.finditer(s): print(i.groupdict())
```

```
{'tagtype': 'w', 'c5': 'NN1', 'form': 'FACTSHEET'}
{'tagtype': 'w', 'c5': 'DTQ', 'form': 'WHAT'}
{'tagtype': 'w', 'c5': 'VBZ', 'form': 'IS'}
{'tagtype': 'w', 'c5': 'NN1', 'form': 'AIDS'}
{'tagtype': 'c', 'c5': 'PUN', 'form': '?'}
```

We can clearly use this to find just the forms (by looking only for the group '**form**') or by finding form: tag pairs (by looking for the values of '**form**' and '**c5**').

The function **reader** will make a reader to scan a directory, looking for words or tagged words in leaf files whose name matches a given pattern (e.g. when scanning through the BNC we just want to look at files whose name ends with .xml, because there are other leaf files in this directory which we do not want to look at: note the pattern is just a standard regular expression, not something which will match Unix filename wildcards). Calling **reader** produces a generator object:

```
>>> bncwordreader = readers.reader(BNC.PATH, readers.BNCWordReader, pat-
tern=".*xml")

>>> bncwordreader
```

```
<generator object reader at 0x7fe2a63d4ac0>
```

We can turn this into a list, but given the size of the BNC this might not be the best thing to do, because loading the entire BNC will produce quite a long list and is likely to take a while:

```
>>> bncwords = list(bncwordreader)
>>> len(bncwords)
```

```
111651731
```

It may therefore make sense just read a smaller initial fragment: for sure, using 1M words for training some algorithm will give us a rough idea of whether it's any good, so we might as well do that first. **listN** will do this for us.

```
>>> bncwordreader = readers.reader(BNC.PATH, BNCWordReader,
pattern=".*xml")
>>> bncwords1000000 = listN(next(bncwordreader), 1000000)
>>> len(bncwords1000000)
```

```
1000000
```

We can also specify that we only want words from one (or more) of the subdirectories:

This will just look at leaf files for which the path contains /B or /C, i.e. files from the B and C subsections of the BNC. This can be useful when splitting the corpus into training and testing portions – you might train an algorithm on everything except these two sections and then test it on these two.

```
>>> bncwordsBC = list(bncwordreaderBC)
>>> len(bncwordsBC)
```

```
27046896
```

We can also get a reader that will produce tagged items (just get the first 1M this time to save time!):

```
>>> bnctaggedwordreader = readers.reader(readers.BNC.PATH, readers.BNC-
TaggedWordReader, pattern=".*xml")
>>> tagged = [next(bnctaggedwordreader) for i in range(1000000)]
>>> readers.printall(tagged[:20])
```

```
('FACTSHEET', 'NN')
('WHAT', 'DT')
('IS', 'VB')
('AIDS', 'NN')
('?', 'PU')
('AIDS', 'NN')
('(', 'PU')
('Acquired', 'VV')
('Immune', 'AJ')
('Deficiency', 'NN')
('Syndrome', 'NN')
(')', 'PU')
('is', 'VB')
('a', 'AT')
```

```
('condition', 'NN')
('caused', 'VV')
('by', 'PR')
('a', 'AT')
('virus', 'NN')
('called', 'VV')
```

There are a number of further parameters that can be tweaked: we will look at those as we need them later on.

UDT Readers

We can use the same machinery for making readers for the UDT. This time a typical sentence looks like

```
1     what  what  PRON  _     PronType=Int,Rel  0     root  _     _
2     is    be    AUX   _     Mood=Ind          1     cop   _     _
3     the   the   DET   _     PronType=Art      4     det   _     _
4     cost  cost  NOUN  _     Number=Sing       1     nsubj _     _
5     of    of    ADP   _     _                 7     case  _     _
```

i.e. each line describes a word, with the second element on a line giving the surface form, the third giving the root form (so the second line above has is as the surface form and be as the root), and the third giving the major tag. We can therefore use the following regeular expression to read lines that contain words and dig out the form and tag:

UDTPattern = re.compile("(?P<COUNTER>\d*)\t(?P<form>\S*)\t(?P<FORM2>\S*)\t(?P<TAG1>\S*)\t(?P<TAG2>\S*)\t(?P<TAG3>\S*)\t(?P<hd>\d*)\t(?P<ROLE>\S*)")

This pattern will match lines that contain ten tab-separated items, where the first and seventh are numbers. The only line that match this in the UDT are indeed descriptions of words, so extracting the values for the group '**form**' will get us the words in the chosen files, and extracting '**form**' and '**TAG1**' will get us the tagged versions.

We can use the pattern to specify for instance that we want the words in all the English datasets:

```
>>> udtwordreader = readers.reader(readers.UDT.PATH, readers.UDTWor-
dReader, pattern=".*En.*.wholething")
>>> udtwords = list(udtwordreader)
>>> len(udtwords)
```

```
426888
```

```
>>> readers.printall(udtwords[:10])
```

```
what
is
the
cost
of
a
```

```
round
trip
flight
from
```

We see that there are 426888 words in all the English datasets in the UDT, and we can see the first ten of them (or whatever we want to see).

We can specify a particular dataset, e.g. the English ATIS dataset (this time we will get the tagged versions):

```
>>> udttaggedwordreader = readers.reader(readers.UDT.PATH, readers.UDT-
TaggedWordReader, pattern=".*UD_English-Atis.*.wholething")
>>> udttagged = list(udttaggedwordreader)
>>> len(udttagged)
```

```
61879
```

```
>>> readers.printall(udttagged[:10])
```

```
('what', 'PRON')
('is', 'AUX')
('the', 'DET')
('cost', 'NOUN')
('of', 'ADP')
('a', 'DET')
('round', 'NOUN')
('trip', 'NOUN')
('flight', 'NOUN')
('from', 'ADP')
```

Other corpora

The same machinery will work for other corpora. You have to write a regular expression that picks out the components that you want, but after that you just use that to make a datafile reader and embed that in a general purpose reader. The PATB (Penn Arabic Treebank), for instance, provides a tagged Arabic corpus. Individual words look like

```
INPUT STRING: تعزيز
LOOK-UP WORD: tEzyz
     Comment:
       INDEX: P1W1
  SOLUTION 1: (taEoziyz) [taEoziyz_1] taEoziyz/NOUN
     (GLOSS): support/backing:strengthening/reinforcing:praise/encourage-
ment/pride
* SOLUTION 2: (taEoziyzu) [taEoziyz_1] taEoziyz/NOUN+u/CASE_DEF_NOM
```

```
     (GLOSS): support/backing:strengthening/reinforcing:praise/encourage-
ment/pride + [def.nom.]

  SOLUTION 3: (taEoziyza) [taEoziyz_1] taEoziyz/NOUN+a/CASE_DEF_ACC

     (GLOSS): support/backing:strengthening/reinforcing:praise/encourage-
ment/pride + [def.acc.]
```

where a number of interpretations are given, with the preferred one in the current case marked with a *.

Given that words in the PATB look like this, the following pattern will find a word and its tag:

```
>>> print(readers.PATBWordPattern.pattern)
```

```
INPUT STRING: (?P<form>\S*).*?\* .*?/(?P<tag>[^/]*)
```

This says that if you look for the words INPUT STRING, then the non-white space that immediately follows is the form. Then look for a * (written as * in the pattern, since * is a significant term in regexes), which will indicate which is the preferred interpretation, and then look for something appearing between two slashes. This pattern will find us words and their tags in the PATB.

```
>>> s = """INPUT STRING: تعزيز

LOOK-UP WORD: tEzyz

     Comment:

       INDEX: P1W1

  SOLUTION 1: (taEoziyz) [taEoziyz_1] taEoziyz/NOUN

     (GLOSS): support/backing:strengthening/reinforcing:praise/encourage-
ment/pride

* SOLUTION 2: (taEoziyzu) [taEoziyz_1] taEoziyz/NOUN+u/CASE_DEF_NOM

     (GLOSS): support/backing:strengthening/reinforcing:praise/encourage-
ment/pride + [def.nom.]

  SOLUTION 3: (taEoziyza) [taEoziyz_1] taEoziyz/NOUN+a/CASE_DEF_ACC

     (GLOSS): support/backing:strengthening/reinforcing:praise/encourage-
ment/pride + [def.acc.]"""

>>> p = readers.PATBWordPattern.search(s)
```

```
('تعزيز', 'NOUN+u')
```

Unfortunately the PATB is not freely available, so we cannot redistribute it here. If you do happen to have it, then you can use our PATBWordReader and PATBTaggedWordReader functions to extract data from it, but the point here is more to note that it is fairly straightforward to write readers for arbitrary datasets so long as you can write a regular expression to match words in the format used by the dataset. Once you have done that you can follow the pattern in **readers.py** for defining a reader: define the regex, embed it in a function for extracting items from leaf files, and use that to make a reader:

```
BNCWordPattern = re.compile("""<(?P<tagtype>w|c) .*?c5="(?P<c5>[A-Z0-
9]*?)(-[A-Z0-9]*)?".*?>(?P<form>\S*?)\s*</(?P=tagtype)>""")


def BNCWordReader(data):

    for i in BNCWordPattern.finditer(open(data).read()):

        form = i.group("form")
```

```
        yield form


def BNCTaggedWordReader(data, N=2):
    for i in BNCWordPattern.finditer(open(data).read()):
        form = i.group("form")
        yield (form, i.group("c5")[:N])


PATBWordPattern = re.compile("INPUT STRING: (?P<form>\S*).*?\* .*?/(?
P<tag>[^/\n]*)", re.DOTALL)


def PATBWordReader(path):
    for i in PATBWordPattern.finditer(open(path).read()):
        yield i.group("form")


def PATBTaggedWordReader(path):
    for i in PATBWordPattern.finditer(open(path).read()):
        yield i.group("form"), i.group("tag")


UDTPattern = re.compile("(?P<COUNTER>\d*)\t(?P<form>\S*)\t(?P<FORM2>\S*)\
t(?P<TAG1>\S*)\t(?P<TAG2>\S*)\t(?P<TAG3>\S*)\t(?P<hd>\d*)\t(?P<ROLE>\
S*)")


def UTDWordReader(data):
    for i in UDTPattern.finditer(open(data).read()):
        yield i.group("form")


def UDTTaggedWordReader(data):
    for i in UDTPattern.finditer(open(data).read()):
        yield (i.group("form"), i.group("TAG1"))


YourWordPattern = re.compile("...(?P<form>...)...(?P<tag>...)")


def YourWordReader(data):
    for i in UDTPattern.finditer(open(data).read()):
        yield i.group("form")


def YourTaggedWordReader(data):
```

```
    for i in UDTPattern.finditer(open(data).read()):
        yield (i.group("form"), i.group("tag"))
```