ASSIGNMENT 1

End-of-Sentence Detection and Text Segment Classification

Due date: Sunday March 1 2020 11:59 PM

Part 1: End-of-Sentence Detection

The goal of the first part of the assignment is to create an algorithm for determining whether a given period (".") in a text indicates an end of sentence or just an abbreviation marker. The following examples illustrate some of the difficulties encountered in this distinction:

```
NOT-END-OF-SENTENCE  119   l viewpoint , as David  C. Robinson has recently shown , t
NOT-END-OF-SENTENCE  128   evealed , '' by Arthur  C. Clarke , Gentry Lee ( Bantam )
NOT-END-OF-SENTENCE  136   E   . The group led by  C. Delores Tucker , head of the NA
NOT-END-OF-SENTENCE  147   ence and Electronics ;  C. Scott Kulicke , on behalf of Se
NOT-END-OF-SENTENCE  184   g Committee chaired by  C. Rubbia . The report covers stat
END-OF-SENTENCE      192    occurs at 440 degrees  C. A hydrogenation test was carrie
END-OF-SENTENCE      210   ystem at 25 and 50 deg  C. Isotherms consist of five branc
END-OF-SENTENCE      239    C while not at 40 deg  C. Minima on the S/sub Pu / vs. C/
NOT-END-OF-SENTENCE  247   ellulases . Culture of  C. thermocellum will be optimized
NOT-END-OF-SENTENCE  255   the cellulase genes of  C. cellulolyticum and those from t
END-OF-SENTENCE      258   anging from 200 to 300  C. A system developed by the autho
END-OF-SENTENCE      262   ourse on programming in C. Finally, those who are interest
END-OF-SENTENCE      300    a house on 2213 Perry Dr. Then the Thomases were seen in
NOT-END-OF-SENTENCE  330     . <P>  Early in 1980 Dr. Thomas B. Reed of SERI and Pro
```

To help you develop a classifier for this distinction, an example set of 45,000 periods and their surrounding context has been provided, each one labelled as `EOS` (End-of-sentence) or `NEOS` (Not-end-of-sentence). The examples were extracted primarily from the Brown Corpus and are located in the sent.data.train on Piazza.

For easy manipulation, the training examples have been divided into tab-delimited columns containing the following information:

- Column 1: `EOS` or `NEOS`, indicating whether the period in that line marks an end of sentence marker or not.
- Column 2: The ID number of the sentence.
- Columns 3-9: The ±3-word surrounding context of the period.
- Column 10: The number of words to the left of the period before the next *reliable* sentence delimiter (e.g. ?, ! or a paragraph marker `<P>`).
- Column 11: The number of words to the right of the period before the next *reliable* sentence delimiter (e.g. ?, ! or a paragraph marker `<P>`).
- Column 12: The number of spaces following the period in the original text.

An example of the first 8 columns for the data above is:

```
TAG   ID#   -3            -2        -1   0      +1              +2
=========================================================================
NEOS  119   as          David     C    .    Robinson       has
NEOS  128   by          Arthur    C    .    Clarke         ,
NEOS  136   led         by        C    .    Delores        Tucker
NEOS  147   electronics ;         C    .    Scott          Kulicke
NEOS  184   chaired     by        C    .    Rubbia         .
EOS   192   440         degrees   C    .    A              hydrogenation
EOS   210   50          deg       C    .    Isotherms      consist
EOS   239   40          deg       C    .    Minima         on
NEOS  247   Culture     of        C    .    thermocellum   will
NEOS  255   genes       of        C    .    cellulolyticum and
EOS   258   to          300       C    .    A              system
EOS   262   programming in        C    .    Finally        ,
EOS   300   2213        Perry     Dr   .    Then           the
NEOS  330   in          1980      Dr   .    Thomas         B
```

The classifier you develop should be able to take data of this format and predict whether the correct label is `EOS` or `NEOS`.

You must empirically derive a decision procedure from the data using a machine learning algorithm such as a decision tree or neural net.

To test the effectiveness of your program, your code will be applied to another file of 5,000 different examples in identical format (`sent.data.test`).

For maximum fairness of the test, you will not be able to see this test data in advance.

To assist you, we have included other files containing wordlists of abbreviations, titles, unlikely proper nouns, and other terms. You may find these useful when building your classifier. To simplify testing and grading, your code must be runnable using the following command:

```
python hw1a.py --train traindata --test testdata --output outputfile
```

where traindata is the path to the training data, testdata is the path to the test fille, and outputfile is a file that will contain the line-by-line classification from your algorithm. We have provided starter code that conforms to this specification and already handles loading the data. In addition, it will compute and print out accuracy on your test data. If your code needs to load any other files, please use relative paths, not absolute paths. We will run your program using Python 3.7.2. If you use any external packages, let us know in your writeup (Part 3) so we can install them if needed. We included two sample classifier implementations. the first one (EOSClassifier1) trains a model to always predict the most common label that it has seen in the training data. If you use sent.train as both the training and test data, using the following command,

```
python hw1a.py --train sent.train --test sent.train
```

youll see that the performance on the training data is over 90%. This reveals that the data is heavily unbalanced. The second classifier (EOSClassifier2) doesnt look at the training data at all! It simply examines the word immediately to the left of the period. If this word is an abbreviation in the provided abbreviations wordlist, then it predicts NEOS, otherwise EOS. This classifier gets 95% accuracy on the training data. Nevertheless, there is still room to improve. However, beware of overfitting. Just because your classifier gets 99.9% accuracy on the training data does not mean it will perform similarly on the test data. As mentioned before, we recommend that you split your training data into a training set and development set and only use the development set for testing. Youll notice that the starter code uses scikit-learn. Feel free to use scikit-learn or your favorite ML package. One benefit of scikit-learn is that you can easily try out many different ML models, and we encourage you to do so. The offcial tutorial can be found at https://scikit-learn.org/stable/tutorial/basic/tutorial.html

Part 2: Text Segment Classification

Much of the text encountered in real-world NLP systems is intermixed with non-textual components such as tables, figures, formulae, and email/netnews headers. Text itself may be standard paragraph style prose or specialized textual segments such as headlines or section headers, addresses, quoted text or email signature blocks. It is useful to distinguish these different segments, both for processing in IR or message routing systems and for obtaining clean prose as training data for language models.

The goal of this part of the assignment is to label each line in a text file with the segment type of the text block the line appears in. For example:

```
NNHEAD  From: desmedt@ruls40.Berkeley.EDU (Koenraad De Smedt)
NNHEAD  Newsgroups: comp.ai.nat-lang
NNHEAD  Subject: CFP: 5th European Workshop on Natural Language Generation
NNHEAD  Date: 24 Oct 1994 09:36:40 GMT
NNHEAD  Organization: Leiden University
NNHEAD  Message-ID: <38fv78$9du@highway.LeidenUniv.nl>
NNHEAD  Keywords: Natural Language Generation Workshop
#BLANK#
HEADL                             CALL FOR PAPERS
#BLANK#
HEADL          5th European Workshop on Natural Language Generation
#BLANK#
HEADL                              20-23 May 1995
HEADL                         Leiden, The Netherlands
#BLANK#
PTEXT This workshop aims to bring together researchers interested in Natural
PTEXT Language Generation from such different perspectives as linguistics,
PTEXT artificial intelligence, psychology, and engineering.  The meeting
PTEXT continues the tradition of a series of workshops held biannually in
PTEXT Europe (Royaumont, 1987; Edinburgh, 1989; Judenstein, 1991 and Pisa,
PTEXT 1993) but open to researchers from all over the world.
#BLANK#
HEADL                           Programme
#BLANK#
PTEXT Papers, posters and demonstrations are invited on original and
PTEXT substantial work related to the automatic generation of natural
PTEXT language, including computer linguistics research, artificial
PTEXT intelligence methods, computer models of human language processing,
#BLANK#
```

```
PTEXT   All contributions should be sent BEFORE 1 JANUARY 1995 to the
PTEXT   Programme Chairman at the following address:
#BLANK#
ADDRESS                Philippe Blache
ADDRESS                2LC - CNRS
ADDRESS                1361 route des Lucioles
ADDRESS                F-06560  Sophia Antipolis
ADDRESS                tel : +33 92.96.73.98
ADDRESS                fax : +33 93.65.29.27
ADDRESS                e-mail : pb@llaor.unice.fr
#BLANK#
#BLANK#
QUOTED  > SJC (San Jose, CA) has an open observation deck on its older terminal
QUOTED  > A.  I have not used this terminal in quite some time, so I don't know
QUOTED  > if sightseers still have access to it.  The problem with this deck was
QUOTED  > that the @@#$^#$%^& restaurant would block your view just as the jets
QUOTED  > were getting off the ground.  Nevertheless, you could still watch the
QUOTED  > planes taxi and then accelerate from the start of the runway.
QUOTED  >
QUOTED  > Why is it that the newer terminals no longer have these outdoor viewing
QUOTED  > areas?  Security, I suppose.  A sad sign of our times.
QUOTED  >
#BLANK#
#BLANK#
SIG                                    ``'''
SIG                                     (0 0)
SIG       +-------------------------00o--(_)--o00-------------------------+
SIG       |                               |                               |
SIG       | Rajeev Agarwal                |  "Hanging in there..."        |
SIG       |                               |                               |
SIG       | Dept. of Computer Science     | (601) 325-8073 or 2756 (Off.) |
SIG       | Mississippi State University  | (601) 325-7506 (Lab)          |
SIG       +-------------------------------+-------------------------------+
SIG                                    (_) (_)
```

A description of the different segment types and examples of them may be found in the Homework directory in the segments subdirectory. Many segments may be classified by the presence of relatively simple patterns within the segment, such as Message-ID: or From: in a netnews header NNHEAD, though other segments may be harder to distinguish. e standards for classification are located in the "standard" file, but dont worry about conforming too closely. Priority will be placed on the creativity and completeness of the segment classifier, not on conforming to any arbitrary definitions of what constitutes a signature or table, for example. We have provided starter code that trains a decision tree classifier. We preprocess each input into a feature vector of four features:

- number of characters
- number of characters after trimming whitespace
- number of words
- 1 or 0 depending if a $>$ is present

With only these four features, this classifier gets 85% accuracy on the training data. This method of manual feature extraction is one possible way to approach this problem.

You may also consider using an algorithm to automatically learn features. Similar to Part 1, we must be able to run your code using the following commands:

```
python hw1b.py --train traindata --test testdata --output outputfile --format line
python hw1b.py --train traindata --test testdata --output outputfile --format segment
```

where the –output switch indicates whether your program should classify by line or by segment.

Part 3: Writeup

Create a short writeup documenting your algorithms and approach for Parts 1 and 2. Describe the models you used, report their performance on the training data, and comment on anything else you found interesting (what did/didn't work, other methods you tried, etc.).

Evaluation:

Submissions will be evaluated as follows:

- 30% - PART 1: Quality, completeness and creativity of algorithm
- 5% - PART 1: Performance on training data
- 20% - PART 1: Performance on independent test data
- 20% - PART 2: Quality, completeness and creativity of algorithm
- 5% - PART 2: Performance on training data
- 15% - PART 2: Performance on independent test data
- 5% - PART 3: Writeup

Submission

We are using Gradescope for submitting assignments. Please submit a compressed file of a folder containing the files for this assignment. Please name follow the naming convention **JHUID_HW1** for the directory. This file should include **all files** that we provide (training data, vocabulary files, etc) as well as your hw1a.py and hw1b.py files.