Assignment E

CSE 517: Natural Language Processing - University of Washington

Winter 2023

Please consult the course website for current information on the due date, the late policy, and any data you need to download for this assignment.

This assignment is designed to advance your understanding of machine translation.

Submit: You will submit your writeup (a pdf), your code, and your output (in the formats described below) via Gradescope. Instructions can be found here.

1 Translation Systems

We provide two files:

- `Train.tsv` (the training data) contains a parallel text: each line contains six tab-separated expressions, with each column expressing the same meaning in a different language. The languages are: Chinese, English, Spanish, Hindi, Japanese, and Norwegian. Text in the same tab-separated column is always in the same language.
- `Test.tsv` (the test data) is in a similar format, but in each row, four of the tab-separated columns are empty, and one of them contains a `?` symbol. The other not-empty column contains an expression in that column's language. For example, the first line of this file contains text in the first column (a Chinese expression) and `?` in the third column (which is Spanish), indicating that this text should be translated into Spanish.

Your task is to build 30 translation systems: one for every ordered pair of the six languages represented in the training data. The design is up to you, and you may use any resources that you can find to complete the task. You must turn in:

1. A file named `Test-output.tsv` which is formatted *exactly* like `Test.tsv`, except that every `?` is replaced by the output of your system in the appropriate language (leave the input intact in its column). Be extremely careful to make sure that you put the output in the correct column! We will use a strict script for measuring exact-match accuracy, and no one will check to see if you made a mistake on your columns. (We strongly recommend that you carve off a subset of the training data and create a development set for yourself.)
2. A one-page writeup describing your approach, including any existing tools or resources that you used.
3. A gzipped tarball containing your code.

Some advice:

- Look at the training data carefully! This may give you ideas about how to build your translation system.

- You are free to use a rule-based approach (e.g., finite-state transducers), a classical statistical approach, a neural approach, or something else (e.g., a semantic parser and a generator for each language). Implementations of various kinds of systems are available online.
- You might want to build 30 separate systems, or you might want to try to build a single system that performs all 30 tasks.

2 Evaluation

We taught in class that automatic scores that give partial credit, like the Bleu score, are often used to evaluate machine translation systems. In this assignment, we will use something much more strict: exact-match accuracy. Why is this a reasonable thing to do in this case?

3 Extra Credit

We provide two files:

- `Train2.tsv` (the training data) contains a parallel text: each line contains two tab-separated expressions that have the same meaning.
- `Test2.tsv` (the test data) is in a similar format, but in each row, the second column contains a `?` symbol.

Your task is to build a translation system based on the training data. The design is up to you, and you may use any resources that you can find to complete the task. You must turn in:

1. A file named `Test2-output.tsv` which is formatted *exactly* like `Test2.tsv`, except that every `?` is replaced by the output of your system in the target language (leave the input intact in its column). Be extremely careful to make sure that you put the output in the correct column! We will use a strict script for measuring accuracy, and no one will check to see if you made a mistake on your columns. (We strongly recommend that you carve off a subset of the training data and create a development set for yourself.)
2. Describe your approach (in the same pdf as the first problem), including any existing tools or resources that you used.
3. A gzipped tarball containing your code.