Final Project: The Challenge

Sarika Halarnakar - shalarn1
Sindhuula Selvaraju - sselvar4

Reordering Challenge Problem 6

In machine translation, it is important to preserve word order because the meaning of a sentence can change depending on word order. Languages differ in many ways based on word order alone. Word order defines relationships between words. Some classifications can be made by naming the typical order of subject, verb, and object, but there are also many other differences in word orders such as Adjective/Noun placement, Noun/Relative Clause placement, and in some languages the word order is not fixed.

For example, compare the difference between

"I had my house cleaned"

and

"I had cleaned my house"

There is an important distinction to make between the two sentences as the word ordering changes the meaning of the sentence.

Reordering is the process of taking input in Spanish:

Se formó un Comité de Selección.

…And finding its best Spanish reordering with the correct English word order under your model:

Un Comité de Selección se formó .

Which would make translating to the English ordered sentence easier:

A Selection Committee was formed.

To reorder, we need a model of reordered Spanish sentences conditioned on the original Spanish sentence. In this assignment, we will give you a basic rule-based reordering system that simply reorders the subject, object, and verb clauses from the original Spanish sentence into SVO order.

Your challenge is to improve our system to account for specific word by word reorderings such as noun/adjective placements, relative clauses, adpositions etc. for the Spanish dev dataset.

You will be judged based on how similar your re-ordering is to the english sentences.

Getting Started

Clone the repository:

```
git clone https://github.com/sindhuula/MT_2016/finalproject.git
```

Install NLTK package by following the instructions in:

```
http://www.nltk.org/install.html
```

Under the `reorder` directory, you now have simple reorderer. Test it out!

```
python reorder.py > data/train.output
```

This creates the file `output` with translations of `data/train.es`. Make sure you give the same name to the output file as the input(.en and .es) files. You can compute the accuracy using `evaluate`.

```
python evaluate.py -t train
```

This command calculates the accuracy of your reordering model by comparing your generated word ordering with the correct word ordering. Each sentence is evaluated out of 100 points. 30% of the accuracy each sentence is awarded for generating an SVO form sentence. We have provided you with preliminary code that already passes this test, however you may feel free to improve our method. The remaining 70% is awarded on a word by word ordering comparison with the correct Spanish reordering.

So for example, if the following sentence is the correct Spanish reordering:

El blanco perro ladra a me

S V O

And the original sentence is:

A me ladra el perro blanco

O V S

And the generated reordering is

El perro blanco ladra a me

S V O

Then the computed accuracy would be 30 for the correct SVO ordering and (4/6)*70 for the word by word ordering, totaling 76.67%.

The training and development sets given to you contain the original Spanish and English sentences (train.es/train.en, dev.es/dev.en). We have also provided a POS tag file for each Spanish dataset (Documentation). A basic English-Spanish dictionary has also been provided, feel free to add data to it.

The Challenge

Your task is to **generate the most probable Spanish reordered sentence with the most probable English word order**.

Our model assumes that the Spanish sentence can be evenly split into subject, verb, and object clauses. This method parses input sentences, groups the words into their respective clauses, and reorders the clauses using a set of hand- crafted rules to get SVO-like sentences.

To pass, you must build on the reorderer we have given you so that it has a complete rule set that is capable of n-gram modelling.

However, rule-based reordering is language specific, so a linguist has to find the best ruleset for every language pair. Though there are successful rulesets for many language pairs, if we could completely reorder the words in input sentences by preprocessing to match the word order of the target language, we would be able to greatly reduce the computational cost of machine translation systems. To get full credit, you **must** additionally experiment with another reordering algorithm.

Any permutation of phrases is a valid translation, so we strongly suggest having a clear training algorithm to model the data. You can use reordering limits as described in the textbook (Chapter 6) and lecture slides. Some things you might try:

- Automatically learning source-side reordering rules
- Pre-ordering of phrase-based machine translation.
- Chunk-Based Verb Reordering.

But the sky's the limit! There are many ways to reorder. You can try anything you want as long as you follow the ground rules:

Ground Rules

- You can work in independently or in groups of up to three, under these conditions:
    1. You must announce the group publicly on piazza.

   2. You agree that everyone in the group will receive the same grade on the assignment.
   3. You can add people or merge groups at any time before the assignment is due. **You cannot drop people from your group once you've added them.** We encourage collaboration, but we will not adjudicate Rashomon-style stories about who did or did not contribute.
- You must turn in three things:
   1. Your reorder result of the entire dataset, uploaded to the leaderboard submission site. You can upload new output as often as you like, up until the assignment deadline.
   2. Your code. Send us a URL from which we can get the code and git revision history (a link to a tarball will suffice, but you're free to send us a github link if you don't mind making your code public). This is due at the deadline: when you upload your final answer, send us the code. You are free to extend the code we provide or roll your own in whatever language you like, but the code should be self-contained, self-documenting, and easy to use.
   3. A clear, mathematical description of your algorithm and its motivation written in scientific style. This needn't be long, but it should be clear enough that one of your fellow students could re-implement it exactly. If you modified your algorithm or have more than 1 algorithm, explain each modification/algorithm clearly. Give the dev scores for each modification/algorithm, and the test score for your final choice.
- You do not need any other data than what we provide. You can free to use any code or software you like, **except for those expressly intended to reorder machine translation models**. You must write your own reorderer. Machine translation software including (but not limited to) Moses, cdec, Joshua, or

phrasal is off-limits. You may of course inspect these systems if it helps you understand how they work. But be warned: they are generally quite complicated because they provide a great deal of other functionality that is not the focus of this assignment. It is possible to complete the assignment with a modest amount of python code. If you aren't sure whether something is permitted, ask us. If you want to do system combination, join forces with your classmates.

- The deadline for the leaderboard is <xyz> at 11:59pm.