Spell Corrector

Egemen Göl
2016400009

Cmpe 493 - Introduction to Information Retrieval
Assignment 1

Implementation

Learn

The spelling corrector starts by learning from corpus.txt which contains assumed-correct English text by tokenizing it with a regular expression and counting every token.

Then using the spell-errors.txt file, the program constructs a dictionary where on every misspelled word there, it notes the corrected word and increments this conversion's count. This is just like the corpus Counter object but it counts for every mis-typed word by itself.

Correct words from spell-errors are added to the corpus, in a weighted manner, because it is assumed that this file is important from the test cases. Current weight is 3, meaning every correct spell-error word is counted three times in the corpus.

Correct

Correction works by asking corpus and spell-errors objects for their correction suggestions, and taking the weighted best among them. Current weight favors spell-error by 30.

If the word exists in the corpus, no correction takes place.

Corpus calculates the probability in two ways.

- If simple mode is selected, probability is defined by (the count of the word in corpus) / (corpus size).
- If smooth mode is selected, probability is defined by (the count of the word in corpus + 1) / (corpus size + 1 * (nof unique words+1))

If there are multiple words with the same probability, for the sake of unbiasedness, the preferred word is selected randomly, in a pre-seeded manner.

1-edit distance is calculated by constructing deletes, transposes, replaces and inserts of every combination and then these are feeded to corpus probability function to calculate the respective frequencies.

Spell-error calculates the probability by choosing the most common corrections, randomly selects one from them, just like corpus.

If they have no suggestions, they return empty string with probability zero. This allows easy choosing between these two suggestions and returning empty string.

Measure

Taking measurements was the heaviest part of the project, since we decided to have measurement mechanism to act as if the correction system is a black box, and build statistics from there.

Two types of statistics are calculated:

1. Confusion matrices
2. Accuracy metrics.

Confusion Matrices

Statistics engine assumes the “correct correction” takes place by suggesting a “1-edit-distance” correction. If it is correct in this suggestion, with respect to the reference word, the respective confusion matrix is updated.

Matrix indexing works like following:

```
For REPLACE x -> y ==> (x, y)
For TRANSPOSE xy -> yx ==> (x, y)
For INSERT x -> xy ==> (x, y)
For DELETE xy -> x ==> (x, y)

For INSERT and DELETE, gives '_' for x if y is at the beginning of the word.
```

Then the confusion alphabet is `'_abcdefghijklmnopqrstuvwxyz'`

Matrice files are in .csv format, with index and header included.

Confusion matrix calculation, since it uses 1-distance editing assumption, only uses the ‘spell-errors.txt’ file as a corpus consisting of only the corrected words, rather than a dictionary of suggestions.

Accuracy Metrics

Accuracy metrics take ‘spell-errors.txt’ file into consideration, since it directly works on the output of the correction system and comparing it to the reference. It tries to find out the method used in correction, and it classifies this method into types:

- EDIT
- TABLE
- NO_OP

EDIT means 1-edit-distance correction, TABLE means the suggestions from the ‘spell-errors.txt’ are used, NO_OP means the word was already correct.

For measuring the ‘spell-errors.txt’ file, reference word is actually a set of words from the file, since it frequently contradicts itself. The set consists of the most-frequent suggestions from the file. If the system corrects the word into any of these reference words, it is counted as a correct correction.

Command-Line Usage

There are two python executables:

1. Correction executable ‘main.py’
2. Measurement executable ‘measure.py’

main.py

```
$ python3 main.py -corpus ./data/corpus.txt -spell-errors ./data/spell-errors.txt simple <
./data/test-words-misspelled.txt > output.txt
```

Uses stdin and stdout for correction. ‘smooth’ or ‘simple’ must be selected by the user.

`-corpus` and `-spell-errors` can be skipped since they default to `./data/corpus.txt` and `./data/spell-errors.txt` files, respectively.

```
$ python3 main.py smooth < ./data/test-words-misspelled.txt > output.txt
```

measure.py

```
$ python3 measure.py all
```

Produces confusion matrix and accuracy metric files into the `./measurements/` folder.

Takes a bit of time, ~15 seconds on the author’s computer.

Measurement Outputs

Confusion Matrices

They can be obtained by running:

```
$ python3 measure.py all
```

Since there are 2*2*4=16 tables, we will include only the insert matrix from simple insertion from the ‘spell-errors’ test set.

	_	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
_	0	0	0	1	0	0	3	1	2	0	0	3	1	2	0	1	2	0	1	7	1	0	1	7	0	0	1
a	4	0	11	71	13	4	8	29	2	100	0	2	288	27	115	0	51	0	123	88	65	34	3	2	0	7	2
b	0	2	0	0	0	9	0	0	0	12	0	0	10	0	0	6	0	0	7	1	1	7	0	0	0	0	0
c	0	15	0	4	0	191	0	0	99	116	0	11	5	0	0	15	0	0	9	21	17	11	0	0	0	15	0
d	4	20	0	0	0	48	0	12	0	46	0	0	8	0	0	2	0	0	5	4	0	7	0	0	0	5	1
e	6	161	0	30	253	28	19	3	0	17	0	1	94	20	130	15	3	0	185	150	51	16	0	7	5	9	11
f	1	3	0	0	0	17	4	0	0	23	0	0	3	0	0	1	0	0	5	0	5	20	0	0	0	0	0
g	0	19	0	0	0	84	0	9	60	18	0	0	2	0	11	1	0	0	7	1	5	22	0	0	0	17	2
h	0	5	0	0	0	55	0	0	0	18	0	0	3	0	6	20	0	0	2	1	32	0	0	0	0	7	2
i	0	97	5	117	22	142	11	23	0	0	0	0	35	28	91	59	33	0	18	91	110	8	6	0	0	0	14
j	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	1	0	0	0	0	0
k	1	0	0	0	0	27	0	0	0	4	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2
l	3	17	0	2	13	197	1	0	0	37	0	1	31	0	0	50	1	0	0	7	4	8	1	0	0	62	9
m	0	30	20	0	0	82	0	0	0	26	0	0	0	2	12	39	19	0	0	6	0	4	0	0	0	0	4
n	8	56	0	43	50	186	0	67	0	76	1	0	4	0	4	13	1	1	0	60	118	15	0	0	2	9	25
o	3	19	3	25	8	11	5	9	0	14	0	1	40	57	86	7	22	0	72	22	22	188	1	30	3	19	2
p	2	16	0	0	0	33	0	0	62	3	0	0	9	0	0	13	5	0	24	0	16	1	0	0	0	2	0
q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0
r	2	80	0	20	10	216	1	1	3	94	0	0	3	3	25	22	2	0	2	24	18	12	0	0	0	45	12
s	3	13	0	37	0	74	1	0	25	82	0	0	0	10	0	17	6	0	0	19	64	4	0	15	0	11	0
t	6	68	0	9	0	369	0	3	46	139	0	0	18	1	1	31	0	0	40	34	11	40	0	1	0	9	17
u	0	34	12	35	11	51	10	12	0	46	0	0	53	32	31	5	26	1	56	50	17	0	0	0	0	0	1
v	0	3	0	0	0	79	0	0	0	10	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0
w	0	3	0	0	1	29	0	0	14	1	0	0	2	0	0	1	0	0	1	1	0	0	0	0	0	1	1
x	0	2	0	13	0	3	0	0	6	10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
y	4	9	0	7	0	14	0	0	0	0	0	0	3	9	0	3	0	0	0	8	0	0	0	0	0	0	4
z	0	3	0	0	0	4	0	0	2	1	0	0	1	0	0	0	0	0	2	12	5	0	0	0	0	0	3

Accuracy Metrics

Test, Simple	Test, Smooth	Spell, Simple	Spell, Smooth

<pre>{
  "true": {
  "edit": 353,
  "table": 12,
  "no_op": 0
  },
  "false": {
  "edit": 13,
  "table": 1,
  "no_op": 5
  },
  "null": 0
}</pre>
	<pre>{
	  "true": {
	  "edit": 353,
	  "table": 12,
	  "no_op": 0
	  },
	  "false": {
	  "edit": 13,
	  "table": 1,
	  "no_op": 5
	  },
	  "null": 0
	}</pre>
		<pre>{
		  "true": {
		  "edit": 22735,
		  "table": 11176,
		  "no_op": 17
		  },
		  "false": {
		  "edit": 879,
		  "table": 0,
		  "no_op": 2316
		  },
		  "null": 0
		}</pre>
			<pre>{
			  "true": {
			  "edit": 22735,
			  "table": 11177,
			  "no_op": 17
			  },
			  "false": {
			  "edit": 878,
			  "table": 0,
			  "no_op": 2316
			  },
			  "null": 0
			}</pre>

~95%	~95%	~91%	~91%

We can observe much from this table.

First, we need to take these percentages with a grain of salt since the system is trained over these datasets, a high percentage is expected.

Second, in our implementation, smoothing makes no difference at all. Even worsens the score in the spell-errors dataset.

“null” means “no-correction since I have no idea”. Our program is trained over these datasets, it always has an idea.

Incorporating the ‘spell-error’ file made little difference, in the test_set the contribution is 12/384 == +3%, not that much. Since other is the training set itself, it is meaningless.

We did not optimize the weight we put into merging ‘spell-errors’ into corpus (3x), since it is not measured. The effect is indirect, hard to measure.

The ‘no_op’ errors in ‘spell-errors’ test is surprising, we are not sure why there are so much errors there. System just leaves the word as is, this may seem like stemming from the weighted merge operation, that move uses only the ‘correct’ or ‘target’ words from that file. Maybe this is because of the inconsistencies within the ‘spell-error’ file.

Maybe the biggest reason for this is the assumed-correct situation in which if the word exists in the corpus (and the corpus includes the target words from the ‘spell-errors’ file’), the system does not correct it. This is a choice of the authors, since user must be respected. Excluding this may improve our percentages, but this is good practice in our judgement.

Criticism and Possible Improvements

As mentioned above, weighted merge operation's effect is unknown.

Also the high error rate from no_op must be investigated further, it is thought that the inconsistency within the 'spell-error' file may be the culprit.

The system is slow, this may be improved by a re-write in a faster language.

Since 1-edit distance correction is the main requirement, other techniques are not discussed.

The keyboard layout for mobile and desktop, incurs some statistical bias to the errors, this may be exploited.

Using a bigger (and more consistent) training set would definitely help.

The choice of not correcting when the word is in the corpus, feels right by the user. There is no need to be a *know-all* in the eyes of the user.

In hindsight, I am satisfied with the approach I took, the architecture I chose and the implementation order I decided. If I would do this assignment once more, there would be few things that I would change. It was a fun and challenging project, especially in terms of testing. Thank you.