Darth Linguo, generating ungrammatical data by corruption.

Creating data to evaluate Language Models on Ungrammatical input.

Pablo Gonzalez Martinez
(@linguomalkavian)

About the author

- Ph.D Student in linguistics at the CUNY Graduate Center
- Working on NLP and Compling
- Keeps finding himself in intersections

What is a Language Model for ? A crash course

- Language models check if the output of a system is a likely string
  - In speech systems: check which sequences of sounds are most likely known words and then which word is most likely etc
  - In Machine Translation systems: check the “Fluency” in the target language

What is a Language Model for ? A crash course 2

- Given a string, a language model scores it with a probability (log probability to avoid underflow)
- A language model is often used to compare outputs, favoring the one with the higher score

Is that English a sentence in?

- Grammaticality refers to whether a given string of words is considered a sentence in the language
- It is usually an intuitionistic notion, used by linguists to pry at the structure of human language

The difference between LM likelihood and grammaticality

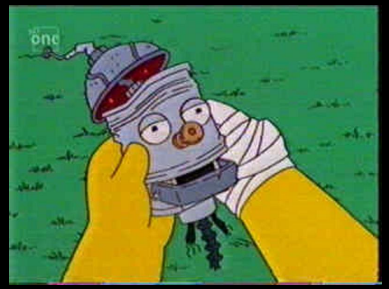
- A language model checks how likely an output is:
  - It returns a number.
  - It usually compares the likelihood of several candidates (relative)
- Grammaticality is a binary measure:
  - A speaker deems a certain string either grammatical or ungrammatical
  - The string is judged by itself (absolute)

In the context of translation

- A human translator generates a sentence that is grammatical in the target language (hopefully) and that captures the meaning of origin sentence.
- An SMT system generates candidates that are the most likely to be a translation (accuracy) and be valid (fluency).

Negative evidence, an old conundrum

- Negative evidence is input that violates the rules of the language
- It is used by linguists to pry at the structure and rules of a language
  - I like oranges
  - I oranges like



But kids don’t use it

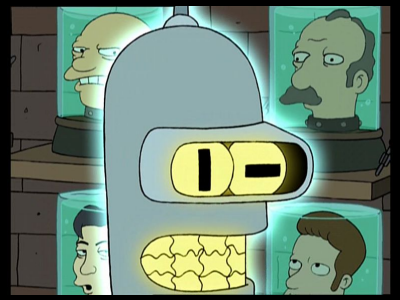
At least according to Chomskians (ask your linguist friends*)

According to part of the poverty of the stimulus hypothesis it is mostly absent from the language exposure children use.

*If you don’t have any you can borrow some of mine

Can a computer benefit from it?

- There is no way to give negative feedback to a traditional N-gram model
- But Neural networks can be trained in a binary classification framework
- Negative reinforcement!



How?

Stay tuned, unfortunately this is not what this year's talk is about.

In short, binary classifiers using convolutional neural networks (fingers crossed) superimposed to traditional NNLMs

Where do we get ungrammatical data?

- Use the errors from our system (Not Here)
  - Learn from the system's mistakes
  - Useful for refining performance
  - Requires human review of the output
- Generate it !
  - We can use what we know about the grammar
  - Useful way for combining approaches into one system

What would the benefit be?

- Informing a statistical system with linguistic information (a mixed approach)
- Possibility for direct control on ungrammatical output for correction
- Possibility to do training on smaller corpora (low resource languages)

Let’s get down to business

- Generate data from faulty structures (use templates)
  - Number and person disagreement between verb and subject (long term dependency)
  - Subject or object absence (structural problem)
- Pool statistical distributions for the sentences from the distributions in the positive data.

Morphology

- Spanish has a rich morphological agreement system (more complex than English)
- We'll start with adjectives

The heavy lifting

- Thank you stanford!


```
Sentence #2 (20 tokens):
Como todos han podido comprobar, el gran "efecto del año 2000" no se ha producido.
[Text=Como CharacterOffsetBegin=227 CharacterOffsetEnd=231 PartOfSpeech=cs NamedEntityTag=O]
[Text=todos CharacterOffsetBegin=232 CharacterOffsetEnd=237 PartOfSpeech=pi000000 NamedEntityTag=O]
[Text=han CharacterOffsetBegin=238 CharacterOffsetEnd=241 PartOfSpeech=vaip000 NamedEntityTag=O]
[Text=podido CharacterOffsetBegin=242 CharacterOffsetEnd=248 PartOfSpeech=vmp0000 NamedEntityTag=O]
[Text=comprobar CharacterOffsetBegin=249 CharacterOffsetEnd=258 PartOfSpeech=vmn0000 NamedEntityTag=O]
[Text=, CharacterOffsetBegin=258 CharacterOffsetEnd=259 PartOfSpeech=fc NamedEntityTag=O]
[Text=el CharacterOffsetBegin=260 CharacterOffsetEnd=262 PartOfSpeech=da0000 NamedEntityTag=O]
[Text=gran CharacterOffsetBegin=263 CharacterOffsetEnd=267 PartOfSpeech=aq0000 NamedEntityTag=O]
[Text=" CharacterOffsetBegin=268 CharacterOffsetEnd=269 PartOfSpeech=fe NamedEntityTag=O]
[Text=efecto CharacterOffsetBegin=269 CharacterOffsetEnd=275 PartOfSpeech=nc0s000 NamedEntityTag=O]
[Text=de CharacterOffsetBegin=276 CharacterOffsetEnd=277 PartOfSpeech=sp000 NamedEntityTag=O]
[Text=el CharacterOffsetBegin=277 CharacterOffsetEnd=279 PartOfSpeech=da0000 NamedEntityTag=O]
```

For more:
https://nlp.stanford.edu/software/spanish-faq.shtml#h

```
(ROOT
  (sentence
    (S
      (sp
        (prep (cs Como))
        (sn
          (grup.nom (pi000000 todos))))
      (grup.verb (vaip000 han)
        (grup.verb (vmp0000 podido)
          (infinitiu (vmn0000 comprobar))))
      (fc ,))
    (sn
      (spec (da0000 el))
      (grup.nom
        (s.a
          (grup.a (aq0000 gran)))
        (fe ") (nc0s000 efecto)
        (sp
          (prep (sp000 de))
          (sn
            (spec (da0000 el))
            (grup.w (nc0s000 año) (w 2000))))
        (fe ")))
    (S
      (neg (rn no)))
    (morfema.pronominal (p0000000 se))
    (grup.verb (vaip000 ha) (vmp0000 producido))
    (fp .)))
```

```
161     def transform(self, sentence):
162         # List to store all codified possible transforms
163         possib = []
164         token_list = sentence.tokens
165         for token in token_list:
166             if token.pos == tree_handler.adj_TAG:
167                 match = AdjInflCorruptor.inf_ADJ_regex.match(token.text)
168                 if match:
169                     root = match.group(1)
170                     infl = match.group(2)
171                     pos_inf = ["a", "o", "as", "os", "es"]
172                     pos_inf.remove(infl)
173                     random.shuffle(pos_inf)
174                     for new_inf in pos_inf:
175                         new_word = root + new_inf
176                         if new_word in lemma_list:
177                             rep = [new_word, token.text]
178                             possib.append(rep)
179         if len(possib) != 0:
180             choice = random.choice(possib)
181             newText = sentence.text.replace(choice[1], choice[0])
182             return newText
183         else:
184             return -1
```

Gender and number disagreement in adjectives (short term dependencies)

Verb number and person agreement
(long-ish) term dependencies

- Verbs in spanish have a wide range of conjugations
- Subject-verb agreement can be long range
- This could throw n-grams off (yey?)

```
58      # Morphological model of the present indicative
59      pres_ind_tag = "vmip000"
60      pres_infl = "(o|es|és|ás|e|emos|éis|en|as|a|amos|áis|an|imos|ís)$"
61      verb_pr_ind_regex = re.compile(root_str + pres_infl)
62
63      # Morphological model of regular past imprefect of the indicative
64      imp_ind_tag = "vmii000"
65      imperf_infl = "((aba|ía)(|s|mos|is|n))$"
66      verb_imp_ind_regex = re.compile(root_str + imperf_infl)
67
68      # Morphological model of regular simple past of the indicative
69      perf_ind_tag = "vmis000"
70      perf_infl = "(é|aste|ó|amos|asteis|aron|í|iste|ió|imos|isteis|ieron)$"
71      verb_ps_ind_regex = re.compile(root_str + perf_infl)
```

Not all tenses are created equal

```
80      def transform(self, sentence):
81          token_list = sentence.tokens
82          possib = []
83          for token in token_list:
84              # Assemble the appropriate replacements
85              # Verb is in present indicative
86              pos_inf = []
87              infl = ""
88              if token.pos == VerbInflCorruptor.pres_ind_tag:
89                  pos_inf = ["o", "es", "és", "ás", "e", "emos", "éis", "en",
90                             "as", "a", "amos", "áis", "an", "imos", "ís"]
91                  v_match = VerbInflCorruptor.verb_pr_ind_regex.match(token.text)
92                  if v_match:
93                      root = v_match.group(1)
94                      infl = v_match.group(2)
95                      if infl[0] == "a" or infl[0] == "á":
96                          for i in pos_inf:
97                              if i[0] != "a" and i[0] != "o" and i[0] != "á":
98                                  pos_inf.remove(i)
99                      else:
100                         for inf in pos_inf:
101                             if inf[0] == "a" or inf[0] == "á":
102                                 pos_inf.remove(inf)
```

If only regexes had a good way to deal with accents

Structural corruption (subject or main verb elimination)

- I owe you this one

9. Is there a Spanish dependency parser?

Not at the moment, unfortunately. We plan to release a Spanish dependency parser (with Universal Dependencies) in the near term.

Images from: https://www.pinterest.com/pin/173529391862257493
And: https://nlp.stanford.edu/software/spanish-faq.shtml#h

Thanks

Questions?