Text generation

Oleksii Piddiachyi • Serhii Brodiuk

Task

- Generate 100 000 texts by 3000-5000 chars text as basis
- Add pictures, tables, headings
- Texts must be 98.5% unique against the Internet and 98% unique against one another
- Correct grammar
- Kill all copywriters
- Try to do it at least with comments, tweets and reviews

Approaches

- Deep learning
- Templates and formal grammars
- SOAs and synonymizer
- All together

Mighty Deep learning

Use LSTM to generate text around a keyword

Pros:

- Power of Deep learning
- No need to study the field
- Easy

Cons:

- Cost

Templates

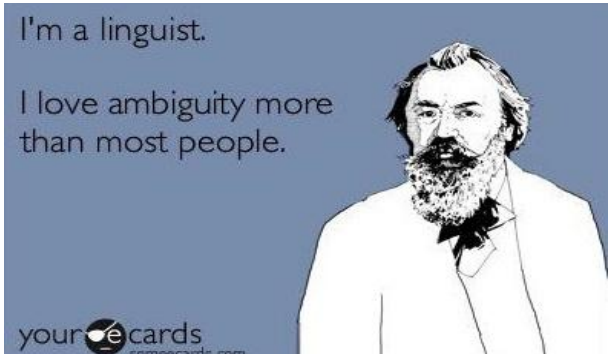
Use templates, formal grammars and rules to construct sentences

Pros:

- Endless improvements
- No need to retrain the model
- Relatively cheap

Cons:

- Must study the field
- Growing complexity



IBM Watson

IBM Bluemix have cool AI - Watson, which can analyse text to extract meta-data from content.

Using Watson’s Language Understanding API we analyse text entities which give us sentences’ keywords.

By saving keywords we keep text close to wanted theme.

Basis and theme

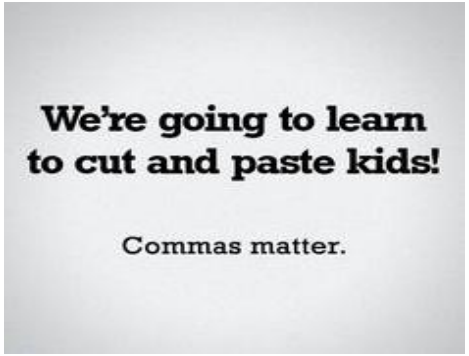
Use SOA of basis texts to change thematic text

Pros:

- No need to retrain the model
- Relatively cheap
- Lots of literature to use
- Can scrobble new texts
- Flabbergasting uniqueness

Cons:

- Must study the field a little
- Growing complexity
- Bad grammar



SOA simple examples

Basis text

- I'd save old Mack if I could.
- I never deal with Zimmerman and Nesbit.

- You'd better tell Simms to attend to it.
- Wait till you see Chicago, though.

Generated text

- world'd suits me if world could.
- world never suits me.

- we'd better accept Simms payments.
- Wait till we accept payments, though.

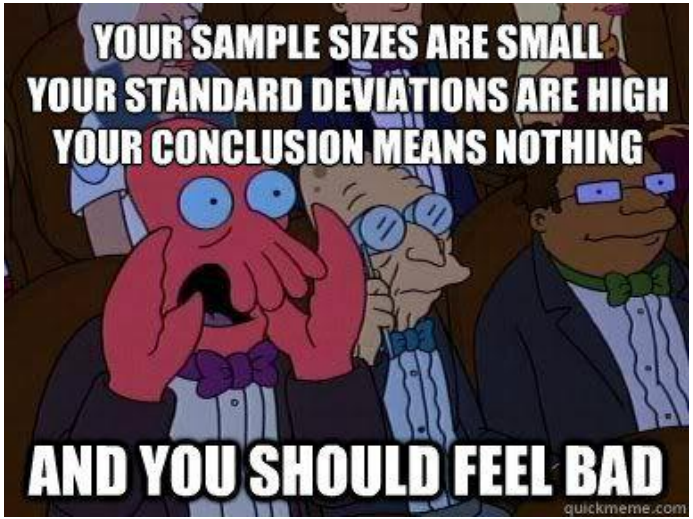
Thematic text

- in one of the most diverse cities in the world also suits me perfectly.

- we accept payments in installments.

Conclusions

- 9/10 generated texts are 100% unique
- Bad grammar
- Plenty of improvement ways
- Whole lot of research is ahead



Problems & Possible Solutions

- Inexact parsing of a syntax tree
  - Clarify the parsing using the **IBM Knowledge Base** and **IBM Watson NLU Service**
  - Use custom models with the **bllip-parser** and **link-grammar**
- Small number of characters at the input
  - Create a list of dynamic rules for expanding topics using **nltk/gensim enrichment rules**
- Errors in the text (readability)
  - Create an architecture from a **group of neural networks**, each of which corrects errors on different scales (**word-form**, **phrase**, **links between phrases**, **sentence**, **links between sentences**)
- The problem of large volume
  - Build a multi-server architecture using **Apache Spark cluster**
  - Create separate services for **Parsing** and **Generating**
- Uniqueness (the problem of a large number of checks)
  - Integration with the verification of uniqueness on a large scale
  - Own semantic service for parsing

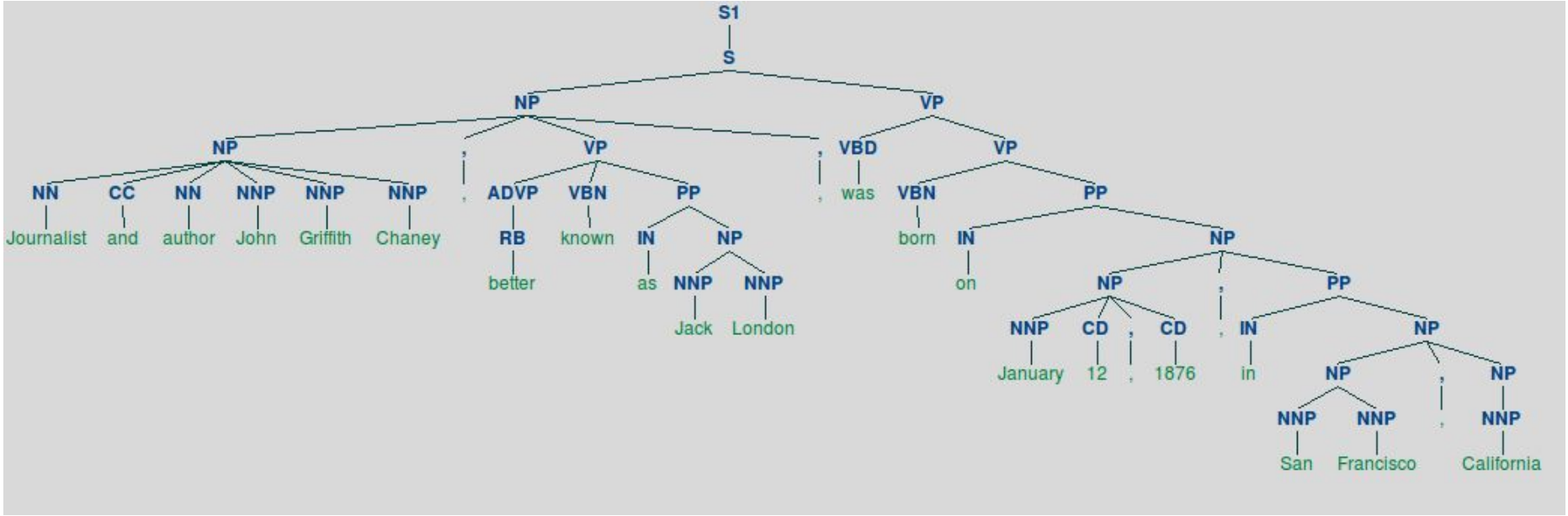
What’s next?

- Smart synonymization
- Syntax trees and formal grammars (bllip-parser, link-grammar)
- Mighty Deep learning
- Parallel computing, using Spark
- New Hope: Try start with comments

Smart Synonymization

- Taking in account the context of word
- Do not change text type by too complicated or too simple synonyms
- Word2vec for searching `close words`

Syntax trees and formal grammars



Result of bllip-parser

Syntax trees and formal grammars

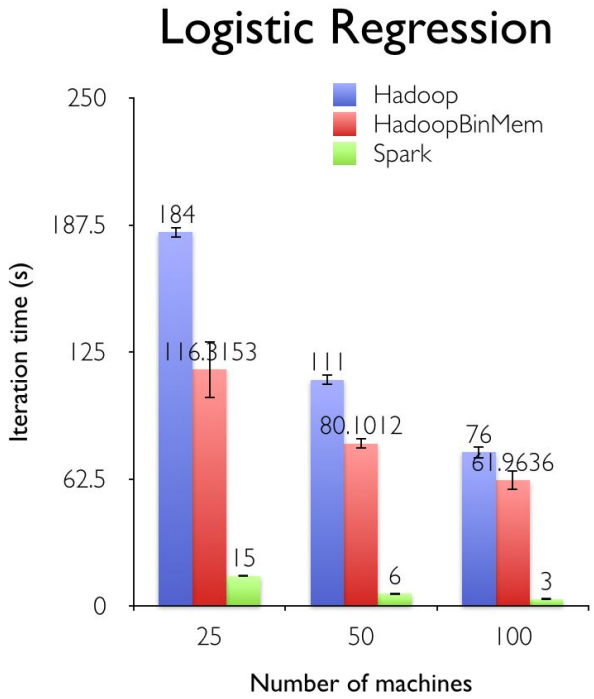
- Chunk extraction
  - e.g. ‘Journalist and author John Griffith Chaney’, ‘Jack London’, ‘January 12, 1876’, ‘San Francisco’, ‘California’
- Grammar parsing
  - DATE: <NNP> <CD> <,> <CD>
  - NAME: <NNP>*
- Using similar tree-structured sentences / phrases for substitution
- Creating rules for combination and substitution:
  - Eg. ADVP VBN should be substituted together

Mighty Deep learning

- RNN
  - sentence correction (Gene Lewis article)
- LSTM
  - text generation
- Word2vec
  - smarter words / phrases substitution
- Markov models
  - text generation
  - phrase completion

Parallel computing with Apache Spark

- Great speed up by automated paralleling
- No need to use to store all data in operating memory help to handle big amount of texts
- Ready libraries for text processing
- It’s updating and became faster with every version



http://emptypipes.org/2015/01/17/python-vs-scala-vs-spark/

Comments, reviews and tweets

- **Positive** vs **Negative**
- **Informative Review** vs **Slang and Local Memes**
- **Pure Information** vs **Jokes and Sarcasms**
- **Product** vs **Service** vs **Thread**
  - **Product Parameters** and **Price Comparison**
  - **Service Review**
    - **Pluses**
    - **Minuses**
  - **Thread Relevance**
- **Information approach**
- **Relevance**
- **No need for unique checks**

Thank you


NOT SURE IF IT'S A DEEP LEARNING JOKE
OR JUST A LACK OF COFFE
memegenerator.net