WG1 Eindhoven

2017-09-26

FTS token auto-suggest using neural networks (Formerly „Deep-Spell“)

Navigation Data Standard

Auto-Suggest FTS token

Background

- Currently token completion suggestion can only be done using the FTS5 vocab table
  - All tokens are stored with their number of occurence there
  - No relations
  - No classification of tokens
- Doing an auto-suggest for tokens based on more than the available information takes a lot of time consuming algorithm and data crunching
- Idea: Let the Artificial Intelligence find out what the best suggestion is

Auto Suggest FTS token

Background

- FTS5 vocab table example
- Even if we would add priorities or other „importance factors“ to tokens we still would need to come up with a certain idea „how“
- Still the tokens are/would not be related to each other → context has no influence on suggestion

Term	Occurences
10th	10.000
new	1000
york	50
main	800
st	100.000
…	…

St L[ouis]	vs	St Loui[siana]	→ Subtoken violation
California Los A[ngeles]	vs	California Los A[lamos]	→ Hierarchy violation
Virginia V[ictoria]	vs	Virginia V[irginia]	→ Redundancy violation

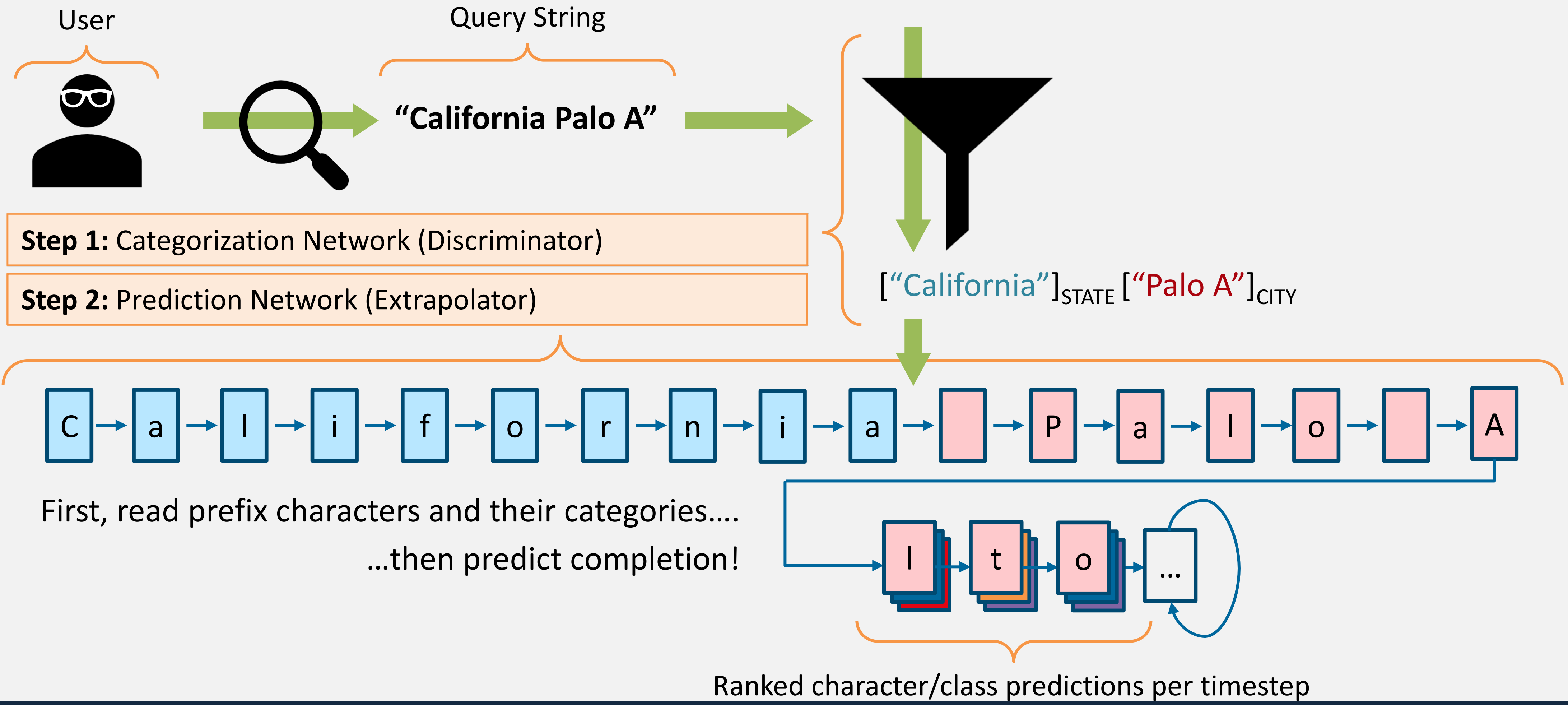
Neural Networks

Solution approach

- Leave query string processing to Neural Network (NN):
  - Tokenization
    - NN input is full user string, not just last term:
      - NN Input: “New York C”
      - Old FTS Input: “C”
  - Categorization
    - NN categorizing hierarchy classes before completion, so actually:
      - User Input: “California Palo A”
      - NN Input: “California Palo A” ➞ [“California”]$_{\text{STATE}}$ [“Palo A”]$_{\text{CITY}}$
  - Suggestion
    - NN returns probability distribution for following characters:
    - suggest([“California”]$_{\text{STATE}}$ [“Palo A”]$_{\text{CITY}}$) = $\left(\begin{bmatrix} l & p = 0.3 \\ n & p = 0.1 \\ \ldots & \ldots \end{bmatrix} \begin{bmatrix} t & p = 0.5 \\ a & p = 0.1 \\ \ldots & \ldots \end{bmatrix} \begin{bmatrix} o & p = 0.3 \\ e & p = 0.2 \\ \ldots & \ldots \end{bmatrix}\right)$

Architecture

Solution approach



User

Query String

“California Palo A”

Step 1: Categorization Network (Discriminator)

Step 2: Prediction Network (Extrapolator)

[“California”]$_{\text{STATE}}$ [“Palo A”]$_{\text{CITY}}$

First, read prefix characters and their categories….

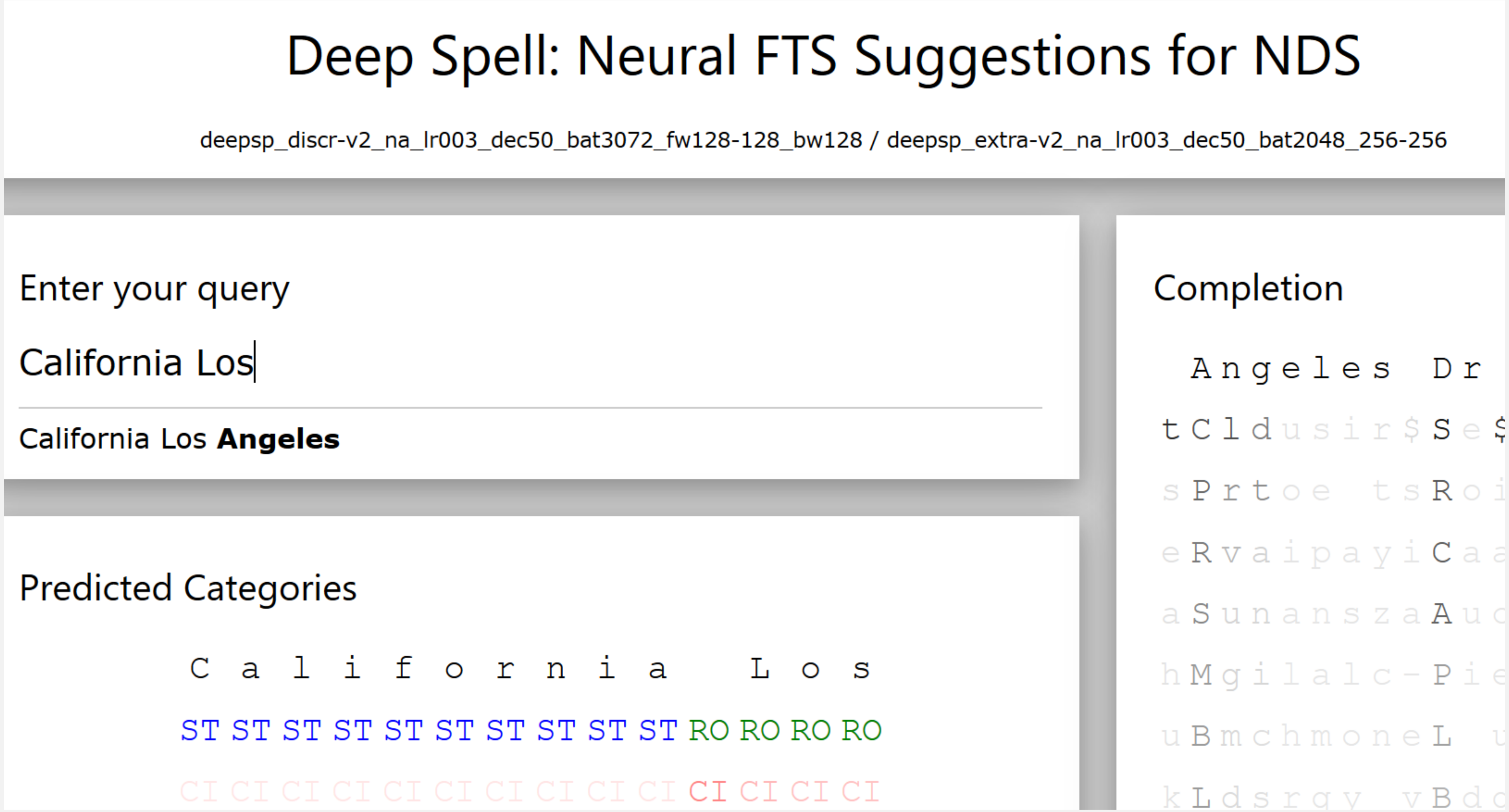
…then predict completion!

Ranked character/class predictions per timestep

Technology stack of demonstrator

- Road FTS North America database with full road coverage
  - Using classification of COUNTRY, STATE, CITY, ROAD
  - About 44,000 cities and total of 9,448,382 roads
- Tensorflow in Python to train neural network models
  - Classifier: 2-way LSTM-RNN (forward, backward), ~ 400K neurons
  - Completer/Suggester: 1 LSTM-RNN, ~600K neurons
  - Trained model size: ~ 10 MB
- Python Webservice to showcase prediction

Demonstrator



Deep Spell: Neural FTS Suggestions for NDS

deepsp_discr-v2_na_lr003_dec50_bat3072_fw128-128_bw128 / deepsp_extra-v2_na_lr003_dec50_bat2048_256-256

Enter your query

California Los

California Los **Angeles**

Predicted Categories

```
C  a  l  i  f  o  r  n  i  a     L  o  s
ST ST ST ST ST ST ST ST ST ST RO RO RO RO
CI CI CI CI CI CI CI CI CI CI CI CI CI CI
```

Completion

```
 A n g e l e s   D r
t C l d u s i r $ S e $
s P r t o e   t s R o i
e R v a i p a y i C a a
a S u n a n s z a A u c
h M g i l a l c - P i e
u B m c h m o n e L   u
k L d s r g y   v B d c
```

Demonstrator Performance

- Running Demonstrator Server on Raspberry Pi 3
- Category prediction and completion wall-time measurement
  - 100-500 ms, depending on prefix length
- Lots of optimization potential
  - 1.2 GHz*4 CPU only, no GPU/TPU acceleration
  - No Neural Network State/Result Caching
  - Python

Demonstrator Performance

Next Steps

- Per-category assessment of completion performance
  - Samples generated by randomly (at least 2 characters prefix) truncating last token postfix from sequence of North American address tokens.
  - Greedy completion precision is measured as percentage of correctly completed characters of last token. **Only respects first completion result -> Greedy!**

Category	Identification-Recall	Identification-Precision	Identification-F1	Greedy Completion Precision
CITY	82%	78%	80%	36%
ROAD	94%	23%	37%	31%
COUNTRY	20%	19%	19%	100%
STATE	84%	65%	73%	97%
ZIP	65%	65%	65%	12%

Demonstrator Performance

Next Steps

Category	Identification-Recall	Identification-Precision	Identification-F1	Greedy Completion Precision
CITY	82%	78%	80%	36%
ROAD	94%	23%	37%	31%
COUNTRY	20%	19%	19%	100%
STATE	84%	65%	73%	97%
ZIP	65%	65%	65%	12%

- Remarks
  - Low Identification of (truncated) country names reflects low presence of countries in randomized addresses -> Network does not expect people to enter countries
  - Low Completion precision of ZIP Codes is not bad (we want the network to generalize address language model)
  - Precision-Recall tradeoff for ROAD class means Network over-expects roads

Open Questions & Possible next steps

- Combine with database content to improve suggestions?
- How to implement an interface for this in NDS?
  - Virtual Table in SQLite?
  - Support different frameworks for DeepLearning?