Deep-learning based Language Understanding and Emotion extractions

Jeongkyu Shin
Lablup Inc.
A sophisticated PaaS that Simplify, Unify and Accelerate processes which enable users to training ML models and execute code on cloud or on-premises with ease.

Labup.AI: Make AI Accessible

**Cloud**

PaaS for research, deep-learning model training and ultra-convenient coding education environment.

**Ground**

Open-source edition for deploying / developing your own Labup.ai Server Farm.

**Garden**

Documents, forum, showcases of Labup.ai platform.

- CodeOnWeb
Usability & extensibility:
- Easier to use
- Extensible to various environment / languages

Scalability:
- More Scalable
- Easier to deploy

MLaaS
- Service scaling
- Computation Runtimes
- Frontends

Sorna

- Apache Spark
- TensorFlow
- Kubernetes
- HIVE
```python
1. import matplotlib.pyplot as plt
2. import pandas as pd
3. 
4. # Read the data into a pandas DataFrame.
6. 
7. # These are the 'Tableau 20' colors as RGB.
8. tableau20 = [(31, 119, 180), (174, 199, 232), (255, 127, 14),
9. (255, 187, 120), (44, 160, 44), (192, 238, 139), (214, 39, 48),
10. (255, 152, 150), (148, 203, 189), (197, 166, 200), (140, 86, 75),
11. (196, 205, 175), (227, 119, 194), (247, 182, 210), (127, 127, 127),
12. (199, 199, 199), (188, 189, 34), (219, 219, 141), (23, 198, 277),
13. (158, 218, 229)]
14. 
15. # Scale the RGB values to the [0, 1] range, which is the format
16. # matplotlib accepts.
17. for i in range(len(tableau20)):
18.     r, g, b = tableau20[i]
19.     tableau20[i] = (r / 255., g / 255., b / 255.)
20. 
21. # Common sizes: (10, 7.5) and (12, 9)
22. plt.figure(figsize=(12, 14))
```
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.backends.backend_sona as sm
import matplotlib
matplotlib.rcParams['svg.fonttype'] = 'none
import matplotlib.pyplot as plt

sm_.backend = 'svg'
plt.close('all')
sns.set(style='ticks')

# Create a dataset with many short random walks
rs = np.random.RandomState(4)
pos = rs.randint(-1, 2, (20, 5)).cumsum(axis=1)
step = np.tile(range(5), 20)
walk = np.repeat(range(20), 5)
df = pd.DataFrame(np.c_[pos.flat, step, walk],
                  columns=['position', 'step', 'walk'])

# Initialize a grid of plots with an Axes for each walk
grid = sns.FacetGrid(df, col='walk', hue='walk', col_wrap=5,
                     size=1.5)

# Draw a horizontal line to show the starting point
```

```
In [2]: import tensorflow as tf

tf.nn.softmax(tf.matmul(x, W) + b)

tf.placeholder(tf.float32, [None, 10])

s_entropy = -tf.reduce_sum(y_ * tf.log(y))

train_step = tf.train.GradientDescentOptimizer(0.01).minimize(s_entropy)

# Session
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

# Learning
for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

# Validation
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

# Result should be approximately 91%.
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
I’m

- Humble business man
  - Lablup Inc.

- Open-source devotee
  - Google Developer Expert (Machine Learning)
  - Textcube open-source project maintainer
    - 10th anniversary!
  - Play with some (open||hidden) projects / companies

- Physicist / Neuroscientist
  - Adj. professor (Dept. of Computer Science, Hanyang Univ.)
  - Ph.D in Statistical Physics (*complex system / neuroscience*)
  - Major in Physics / Computer Science
Today’s focus

- NLP and Sentiment: Big problems when making chatbots
- Natural Language Understanding
  - SyntaxNet and DRAGAN
- Emotion reading
  - SentiWordNet and SentiSpace[1]

[1] Our own definition for sentimental state space
Understanding Language:

It’s even hard for human beings.
Chat-bots with Machine Learning

Today’s focus!

Lexical Input

Natural Language Processor

Sentence To vector converter

Context Analyzer

Decision maker

Response Generator

Lexical Output

Deep-learning model

SyntaxNet / NLU (Natural Language Understanding)

Deep-learning model (RNN / sentence-to-sentence)

Knowledgebase (useful with TF/IDF ask bots)

Per-user context memory
Understanding Languages

- The structure of language
  - “Noun” and “Verb”

- “Context”
  - POS (Part-of-speech)
    - Roles for the words
    - Added as tags
    - Only one meaning in the current sentence context
  - Generalized POS tags
    - Some POS tags are very common (noun, verb, ⋮)
    - Others? Quite complicated!
SyntaxNet (2016)

- Transition-based framework for natural language processing
  - Feature extraction
  - Representing annotated data
  - Evaluation

- End-to-end implementation using deep learning
  - No language-awareness/dependencies: data-driven

- Interesting points
  - Found general graph structure between different human languages (2016-7)
  - http://universaldependencies.org
DRAGNN (2017)

- Dynamic Recurrent Acyclic Graphical Neural Networks (Mar. 2017)
  - Framework for building **multi-task, fully dynamically constructed computation graphs**
  - Not GAN (Generative Adversarial Network)!

- Supports
  - Training and evaluating models
  - Pre-trained analyze models (McParsey) for 40 language
    - Except Korean. (of course; )

*Kong et al., (2017)*
TBRU

- Transition-based recurrent unit
  - Discrete state dynamics: allow network connections to be built dynamically as a function of intermediate activations
- Potential of TBRU: extension and combination
  - Sequence-to-sequence
  - Attention mechanisms
  - Recursive tree-structured models

*Kong et al., (2017)*
Generating NLP with SyntaxNet

1. Obtaining Data
2. POS Tagging
3. Training SyntaxNet POS tagger
4. Dependency parsing Transition-based Parsing
5. Training Parser
SyntaxNet implementation

- Not a BOW (Bag-of-words) model
- Workflow
  - POS Tagging model
  - Preprocessing with tagger model
  - Dependency parsing

```
Input/TransitionState (C++)
Sentence w/ partial annotations

Parsing:
  Stack
  Buffer

Tagging:
  PRP VBD DET
  Stack
  Buffer

Sparse feature extraction

Feed-Forward SyntaxNet Architecture (Overview)
```

github.com/tensorflow/tensorflow
SyntaxNet implementation

- Transition-based dependency parser
  - SHIFT, LEFT ARC, RIGHT ARC
- “deviation”
  - Configuration+Action
- Training
  - Local pre-training / global training

*Kong et al., (2017)*
Dive into TBRU

- TBRU schematic
- Arc-standard transition
  - Choose the right candidate

**Figure 3:** Left: TBRU schematic. Right: Dependency parsing example. A gold parse tree and an initial encoding vector.

Fixed embedding vector for the output tag transition system. For input we call

Example 3. "Input" transducer TBRUs via no-...
Model differences

▪ DRAGNN[1]: End-to-end, deep recurrent models
  ▪ Use to extend SyntaxNet[2] to be end-to-end deep learning model
 ▪ **TBRU**: Transition-Based Recurrent Unit
  ▪ Uses both encoder and decoder
 ▪ TBRU-based multi-task learning: DRAGNN

▪ SyntaxNet: Transition-based NLP
  ▪ Can train SyntaxNet using DRAGNN framework

Parsey McParseface

- Parsey McParseface (2017)
  - State-of-art deep learning-based text parser

Performance comparison

<table>
<thead>
<tr>
<th>Model</th>
<th>News</th>
<th>Web</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ling et al. (2015)</td>
<td>97.44</td>
<td>94.03</td>
<td>96.18</td>
</tr>
<tr>
<td>Andor et al. (2016)*</td>
<td>97.77</td>
<td>94.80</td>
<td>96.86</td>
</tr>
<tr>
<td>Parsey McParseface</td>
<td>97.52</td>
<td>94.24</td>
<td>96.45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>News</th>
<th>Web</th>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Martins et al. (2013)</td>
<td>93.10</td>
<td>88.23</td>
<td>94.21</td>
</tr>
<tr>
<td>Zhang and McDonald (2014)</td>
<td>93.32</td>
<td>88.65</td>
<td>93.37</td>
</tr>
<tr>
<td>Weiss et al. (2015)</td>
<td>93.91</td>
<td>89.29</td>
<td>94.17</td>
</tr>
<tr>
<td>Andor et al. (2016)*</td>
<td>94.44</td>
<td>90.17</td>
<td>95.40</td>
</tr>
<tr>
<td>Parsey McParseface</td>
<td>94.15</td>
<td>89.08</td>
<td>94.77</td>
</tr>
</tbody>
</table>

POS (part-of-speech) tagging

For different language domains

*github.com/tensorflow/tensorflow
McParseface model / DRAGNN framework

ParseySaurus analysis:

Глокая куздра штеко будланула бокра и курдячит бокрёнка

Dynamically constructed network:

Character-based word representations

POS and Syntax

*github.com/tensorflow/tensorflow*
SyntaxNet Architecture

Training with Beam Search:

Sum scores of all decisions across entire history....

Incorrect parse

The horse raced past the barn fell .

Correct parse

The horse raced past the barn fell .

Update: maximize $P$ (correct parse) relative to the set of alternatives

Globally Normalized SyntaxNet Architecture (Overview)

*github.com/tensorflow/tensorflow
DRAGNN implementation

- DRAGNN implementation on TensorFlow
Compute Graph

- Compute graph for SyntaxNet
  - Example case in TensorFlow repo.
- Three parts of NLP
  - Lookahead
  - Tagger
  - Parser
- Characteristics
  - Every model uses memory effect
Why no Korean?

- Korean language-specific characteristics

- Solution?
  - Yes, I think. (testing now.)
Now, let’s move to the emotion part.

Looks easier but harder, in fact.
Problems for next-gen chatbots

- Hooray! Deep-learning based chat bots works well with Q&A scenario!

- General problems
  - Inhuman: restricted for model training sets
  - Cannot "start" conversation
  - Cannot handle continuous conversational context and its changes

- "Uncanny Valley"
  - Inhuman speech / conversation.
  - Why? How?
Emotion engine

Today's focus!

Disintegrator

Deep-learning model
(sentence-to-sentence + context-aware word generator)

Knowledge engine

Context parser

Emotion engine

Context memory

Sentence generator

Grammar generator

Tone generator

Response generator

Lexical Input

NLP + StV

Context analyzer+Decision maker

Lexical Output
Conversational context locator

- Using Skip-gram and bidirectional 1-gram distribution in recent text
- I ate miso soup this morning. => Disintegrate first
- Bidirectional 1-gram set (reversible trigram): \{\langle I, \text{miso soup}, \text{Eat} \rangle, \langle \text{eat, today}, \text{miso soup} \rangle, \langle \text{miso soup, morning}, \text{today} \rangle\}
- Simplifying: \{\langle I \rangle, \langle \text{FOOD} \rangle, \langle \text{EAT} \rangle \}, \{\langle \text{EAT}, \text{Today} \rangle, \langle \text{FOOD} \rangle \}, \{\langle \text{FOOD}, \text{morning} \rangle, \text{Today} \rangle\}
- Distribution: more simplification is needed
  - \{\langle I \rangle, \langle \text{FOOD} \rangle, \langle \text{EAT} \rangle \}, \{\langle \text{TIME:DATE}, \langle \text{EAT} \rangle \rangle, \langle \text{FOOD} \rangle \}, \{\langle \text{FOOD}, \langle \text{TIME:DAY} \rangle \rangle, \langle \text{TIME:DATE} \rangle \}
  - Now we can calculate multinomial distribution

*I'll use trigram as abbreviation of reversible trigram*
Conversational context locator

- Using Skip-gram and bidirectional 1-gram distribution in recent text

나는 오늘 아침에 된장국을 먹었습니다. => Disintegrate first

- Bidirectional 1-gram set: 
  
  - {{나, 아침}, 오늘}, {{오늘, 된장국}, 아침}, {(아침, 멕다), 된장국}

- Simplifying: 
  
  - {{<I>, 아침}, 오늘}, {{오늘, <FOOD>}, 아침}, {(아침, <EAT>), <FOOD>}

- Distribution: more simplification is needed
  

  - Now we can calculate multinomial distribution
Conversational context locator

- Training context space
  - Context-marked sentences (>20000)
  - Context: LIFE / CHITCHAT / SCIENCE / TASK
  - Prepare Generated trigram sets with context bit
  - Train RNN with 1-gram-2-vec

- Matching context space
  - Input trigram sequence to context space
  - Take the dominator axis

- Using Skip-gram and trigram distribution in recent text
  - {(<I>,<TIME:DAY>), <TIME:DATE>}
  - {(<TIME:DATE>,<FOOD>), <TIME:DAY>}
  - {(<TIME:DAY>,<EAT>),<FOOD>}

- With distribution
  - Calculate maximum likelihood significance and get significant n-grams
  - Uses last 5 sentences
For better performance

- Characteristics of Korean Language
  - Distance between words: **important**
  - Sequence between words: **not important**
  - Different from English

- How to read more contextual information from longer text? (e.g. Documents)

- Change from trigram to in-range tri pairs

- I ate miso soup this morning:

  <I> <EAT> <FOOD> <TIME:DATE> <TIME:DAY>

  - In range 1: {(<I>,<FOOD>), <EAT>}
  - In range 2: {(<TIME:DATE>), <EAT>}
  - In range 3: {(<TIME:DAY>), <EAT>}

- Heavily depends on the length of original sentence
  - Short?
  - Long?
Emotion engine

- **Input:** text sequence
- **Output:** Emotion flag (6-type / 3bit)
- **Training set**
  - Sentences with 6-type categorized emotion
    - Positivity (2), negativity (2), objectivity (2)
  - Uses **senti-word-net** to extract emotion
  - 6-axis emotion space by using **Word2Vec** model
- **Current emotion indicator**: the most weighted emotion axis using **Word2Vec** model
  - Position in senti-space: [0.95, 0.05, 0.11, 0.89, 0.92, 0.08]
  - Index: 1 2 3 4 5 6
  - Index: 0x01

Illustration *(c) http://ontotext.fbk.eu/*
Making emotional context locator

- Similar to conversational context locator
  - Just use 1-gram from input
  - Add the corresponding word vector on emotion space

- How to?
  - Use NLTK python library
    - NLTK has corpora / data for SentiWordNet
    - Also gives download option!
Making emotional context locator

- Get emotional flag from sentence

Sample test routine for Sentimental state

```python
from nltk.corpus import sentiwordnet as swn

def get_senti_vector(sentence, pos=None):
    result = dict()
    for s in sentence.split(' '):
        if s not in result.keys():
            senti = list(swn.senti_synsets(s.lower(), pos))
            if len(senti) > 0:
                mostS = senti[0]
                result[s] = [mostS.pos_score(), 1.0 - mostS.pos_score(), mostS.neg_score(), 1.0 - mostS.neg_score(), mostS.obj_score(), 1.0 - mostS.obj_score()]

    return result
```

```python
sentence = "Hello I am happy I was super surprised"
result = get_senti_vector(sentence)
```

Adj. only

```python
{'I': [0.0, 1.0, 0.25, 0.75, 0.75, 0.25],
'happy': [0.875, 0.125, 0.0, 1.0, 0.125, 0.875],
'super': [0.625, 0.375, 0.0, 1.0, 0.375, 0.625],
'surprised': [0.125, 0.875, 0.25, 0.75, 0.625, 0.375]}
```

All morpheme

```python
{'Hello': [0.0, 1.0, 0.0, 1.0, 1.0, 0.0],
'I': [0.0, 1.0, 0.0, 1.0, 1.0, 0.0],
'am': [0.0, 1.0, 0.0, 1.0, 1.0, 0.0],
'happy': [0.875, 0.125, 0.0, 1.0, 0.125, 0.875],
'was': [0.0, 1.0, 0.0, 1.0, 1.0, 0.0],
'super': [0.0, 1.0, 0.0, 1.0, 1.0, 0.0],
'surprised': [0.125, 0.875, 0.0, 1.0, 0.875, 0.125]}
```
Creating Korean SentiWordNet

- Procedure to generate Korean SentiWordNet corpus

1. Get every synsets from sentiwordnet data

```python
for i in swn.all_senti_synsets():
    data.append(i)
```

2. Translate words into Korean

3. Treat synonym

4. Choose the score from ‘representative word’

```python
<불구의.s.01: PosScore=0.0 NegScore=0.0>
<알맞다.a.01: PosScore=0.5 NegScore=0.0>
<적합하다.a.01: PosScore=0.5 NegScore=0.0>
<어울리다.a.01: PosScore=0.5 NegScore=0.0>
<만족스럽다.s.04: PosScore=0.25 NegScore=0.0>
<적합하다.s.01: PosScore=0.125 NegScore=0.0>
<훌륭하다.s.03: PosScore=0.875 NegScore=0.0>
<부적합하다.a.01: PosScore=0.25 NegScore=0.0>
```
Reading emotion with SentimentSpace

- Creating emotion space
  - 1. Generate word space using word2vec model
  - 2. Substitute word to SentiWordNet set
  - 3. Now we get SentimentSpace!
  - 4. Get the emotion state by giving disintegrated word set into SentimentSpace

- Focuses on reading emotion
  - Final location on WordVec space = Average sentivector of nearest neighbors

*SentimentSpace: our definition / approach to simulate emotion.*
Tips for SentimentSpace

- When picking the best match from candidates
  - e.g. fit →
    - <fit.a.01: PosScore=0.5 NegScore=0.0>
    - <acceptable.s.04: PosScore=0.25 NegScore=0.0>
    - <suitable.s.01: PosScore=0.125 NegScore=0.0>
    - <worthy.s.03: PosScore=0.875 NegScore=0.0>

- 1. Just pick the first candidate from senti sets
- 2. Calc the average Pos/Neg scores - [0.25, 0]

- When generating Korean SentiWordNet corpus
  - 1. Do not believe the result. You will need tremendous amount of pre / postprocessing
  - *SentimentSpace* is very rough. Keep in mind to model the emotion engine
Summary

▪ Today
  ▪ Dive into SyntaxNet and DRAGNN
  ▪ Emotion reading procedure using SentiWordNet and deep learning

▪ My contributions / insight to you
  ▪ Dodging Korean-specific problems when using SyntaxNet
  ▪ My own emotion reading / simulation algorithm
Thank you for listening :) 

@inureyes /
fb/jeongkyu.shin

lablup.com