20170627 쉽게 배우는 인공지능

Deep-learning based **Language Understanding** and **Emotion extractions**

BETA

Jeongkyu Shin
Lablup Inc.

KossLab
KOREA OPEN SOURCE SOFTWARE DEVELOPERS LAB
Needlworks
한양대학교
HANYANG UNIVERSITY

Lablup.AI: Make AI Accessible

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.

Cloud

Pay-as-you-Go

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

Ground

Bring Your Own Hardware

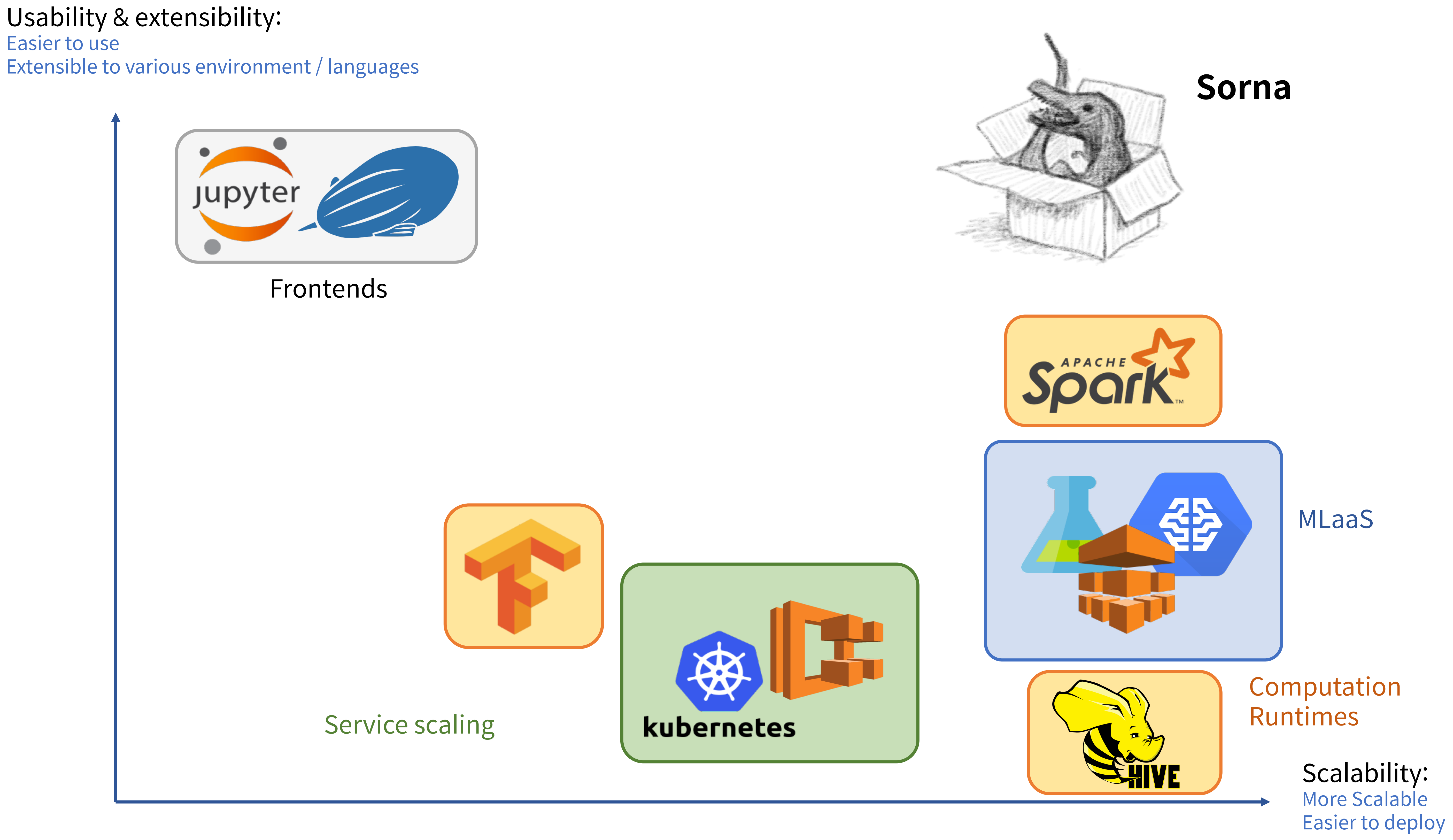
Open-source edition for deploying / developing your own Lablup.ai Server Farm.

Garden

Showcases

Documents, forum, showcases of Lablup.ai platform.

- Code**On**Web


Usability & extensibility:
Easier to use
Extensible to various environment / languages
Sorna
jupyter
Frontends
APACHE Spark
MLaaS
Service scaling
kubernetes
Computation Runtimes
HIVE
Scalability:
More Scalable
Easier to deploy

codeonweb.com

CodeOnWeb - 실습

CodeOnWeb

Lablup Tutorials

메뉴 | 실습 | Untitled ×

대시보드 > 실습

Languages / Environments

python3

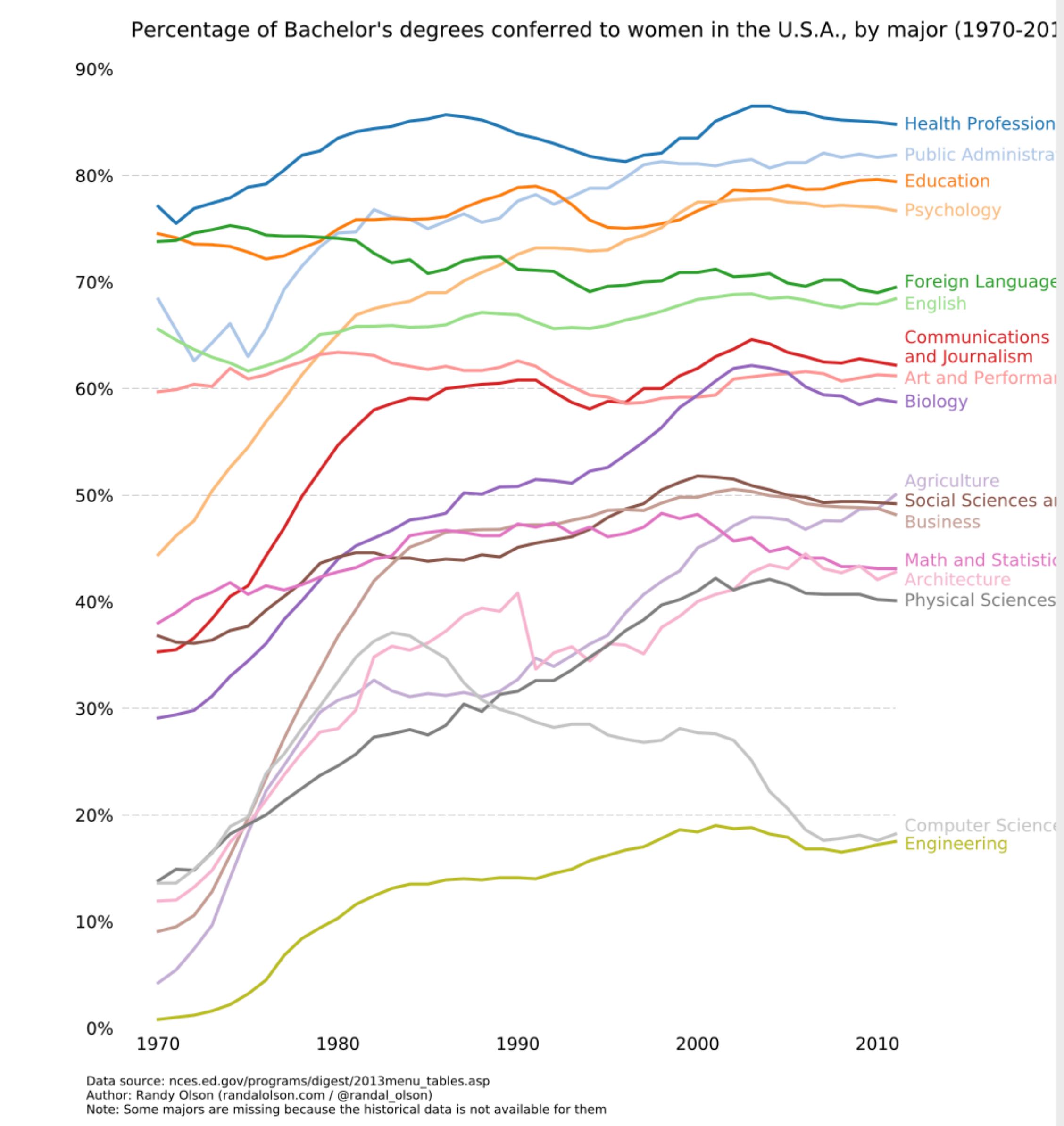
실행

저장

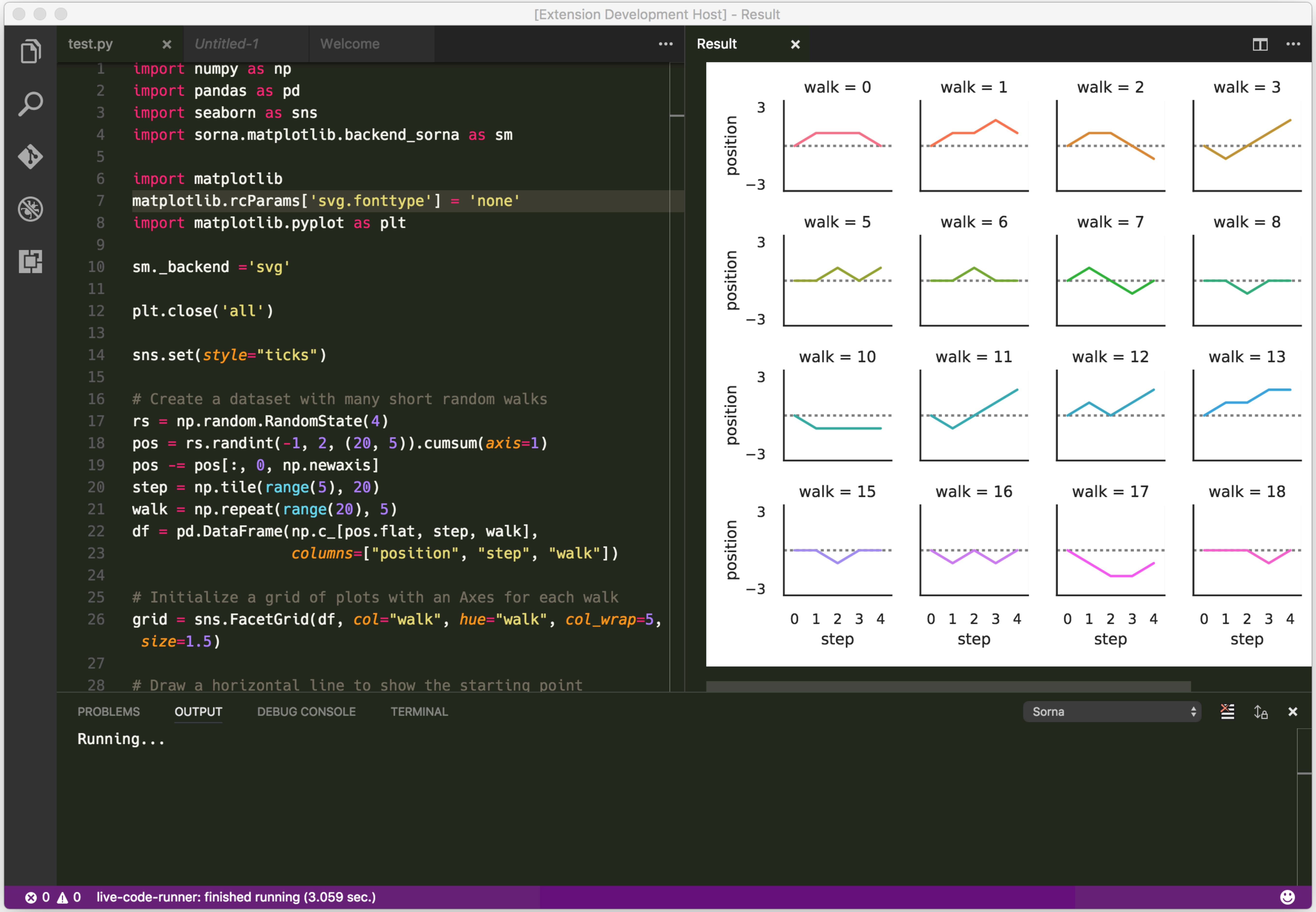
MORE...

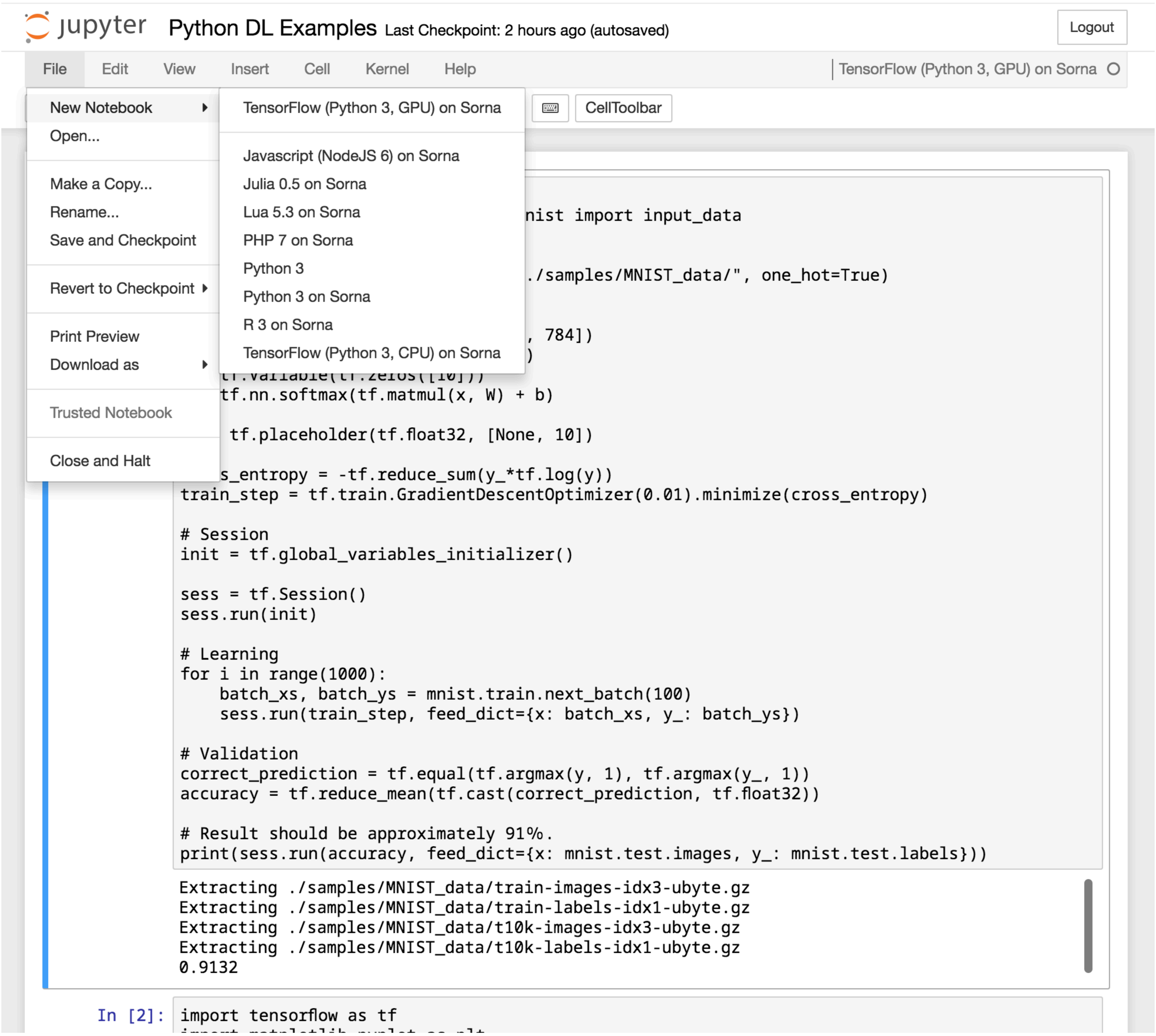
문제 보고 · 사용조건 · 개인정보보호

```
import matplotlib.pyplot as plt
import pandas as pd

# Read the data into a pandas DataFrame.
gender_degree_data =
pd.read_csv("http://www.randalolson.com/wp-
content/uploads/percent-bachelors-degrees-women-usa.csv")

# These are the "Tableau 20" colors as RGB.
tableau20 = [(31, 119, 180), (174, 199, 232), (255, 127, 14),
(255, 187, 120),
             (44, 160, 44), (152, 223, 138), (214, 39, 40),
(255, 152, 150),
             (148, 103, 189), (197, 176, 213), (140, 86, 75),
(196, 156, 148),
             (227, 119, 194), (247, 182, 210), (127, 127, 127),
(199, 199, 199),
             (188, 189, 34), (219, 219, 141), (23, 190, 207),
(158, 218, 229)]

# Scale the RGB values to the [0, 1] range, which is the format
matplotlib accepts.
for i in range(len(tableau20)):
    r, g, b = tableau20[i]
    tableau20[i] = (r / 255., g / 255., b / 255.)

# You typically want your plot to be ~1.33x wider than tall.
This plot is a rare
# exception because of the number of lines being plotted on it.

# Common sizes: (10, 7.5) and (12, 9)
plt.figure(figsize=(12, 14))
```



8.4.6388.20170210.tricera

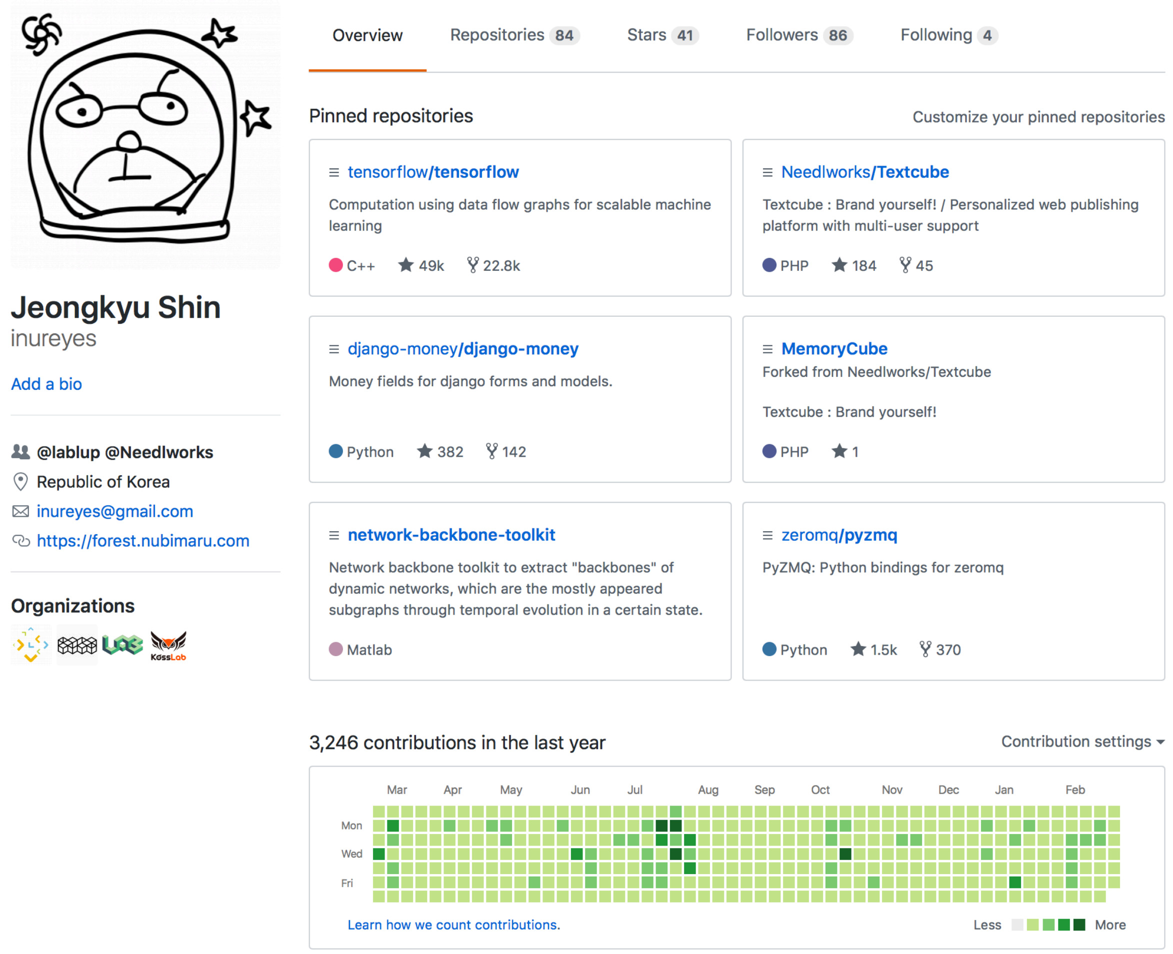

[Extension Development Host] - Result
test.py
Untitled-1
Welcome
import numpy as np
import pandas as pd
import seaborn as sns
import sorna.matplotlib.backend_sorna as sm
import matplotlib
matplotlib.rcParams['svg.fonttype'] = 'none'
import matplotlib.pyplot as plt
sm._backend ='svg'
plt.close('all')
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)
pos -= pos[:, 0, np.newaxis]
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
Result
walk = 0
walk = 1
walk = 2
walk = 3
walk = 5
walk = 6
walk = 7
walk = 8
walk = 10
walk = 11
walk = 12
walk = 13
walk = 15
walk = 16
walk = 17
walk = 18
position
3
−3
0 1 2 3 4
step
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Sorna
Running...
0 0 live-code-runner: finished running (3.059 sec.)


jupyter
Python DL Examples Last Checkpoint: 2 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
TensorFlow (Python 3, GPU) on Sorna
New Notebook
Open...
Make a Copy...
Rename...
Save and Checkpoint
Revert to Checkpoint
Print Preview
Download as
Trusted Notebook
Close and Halt
TensorFlow (Python 3, GPU) on Sorna
Javascript (NodeJS 6) on Sorna
Julia 0.5 on Sorna
Lua 5.3 on Sorna
PHP 7 on Sorna
Python 3
Python 3 on Sorna
R 3 on Sorna
TensorFlow (Python 3, CPU) on Sorna
CellToolbar
nist import input_data
./samples/MNIST_data/", one_hot=True)
, 784])
)
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(cross_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}))
Extracting ./samples/MNIST_data/train-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/train-labels-idx1-ubyte.gz
Extracting ./samples/MNIST_data/t10k-images-idx3-ubyte.gz
Extracting ./samples/MNIST_data/t10k-labels-idx1-ubyte.gz
0.9132
In [2]: import tensorflow as tf

I'm

신정규 / Jeongkyu Shin / @inureyes

- 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

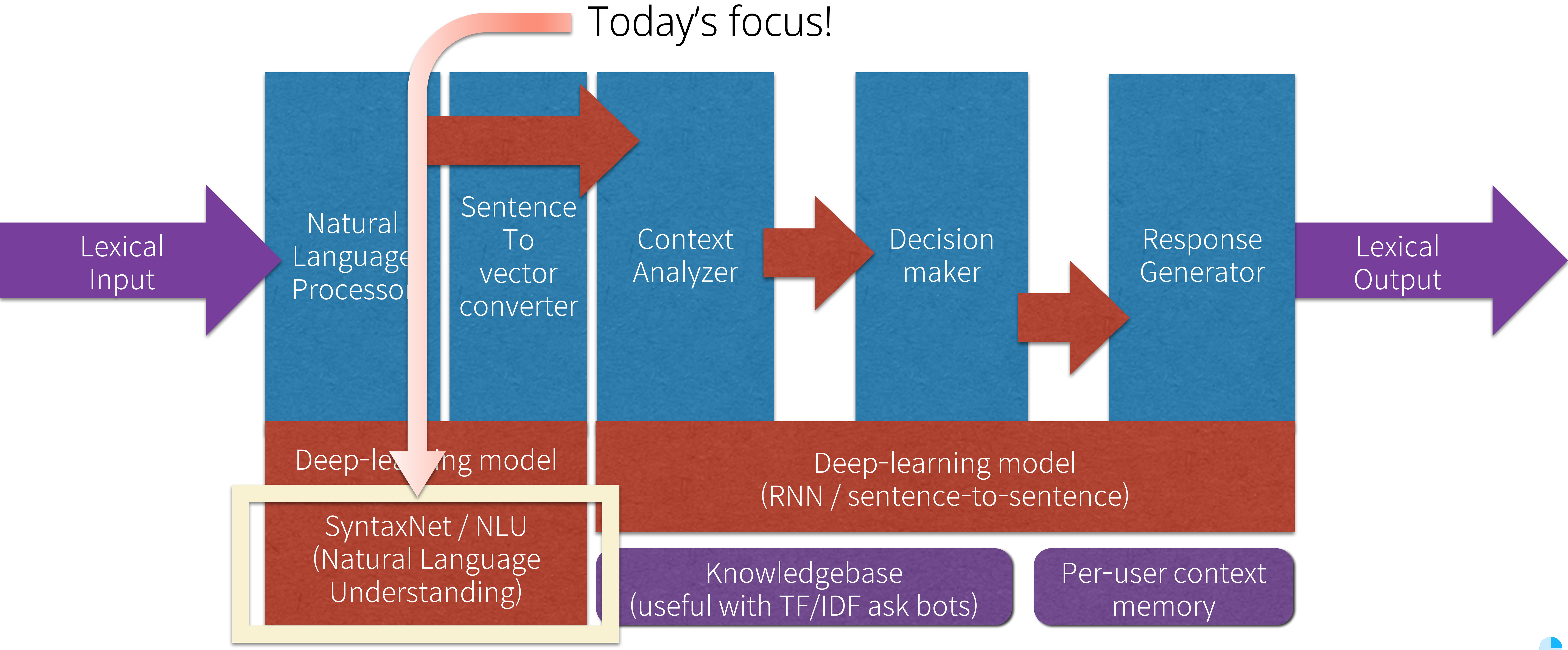
- NLP and Sentiment: Big problems when making chatbots
- Natural Language Understanding
  - SyntaxNet and DRAGAN
- Emotion reading
  - SentiWordNet and SentiSpace[1]

[1] Our own definition for sentimental state space

Understanding Language:

It’s even hard for human beings.

Chat-bots with Machine Learning



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!

ㄱ ㄴ ㄷ

아버지가방에
들어가신다

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

*github.com/tensorflow/tensorflow

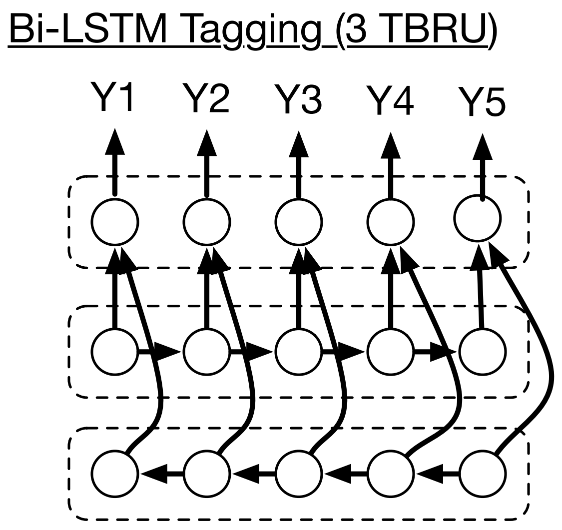
DRAGNN (2017)

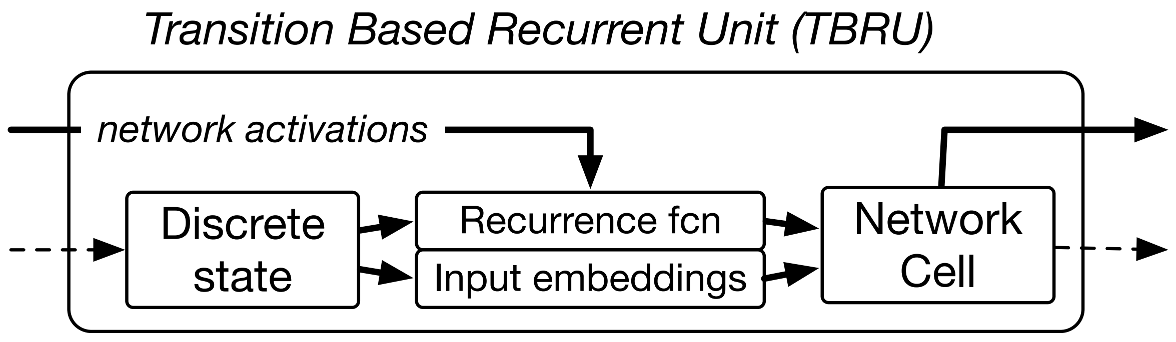
- Dynamic Recurrent Acyclic Graphical Neural Networks (Mar. 2017)
  - Framework for building **multi-task**, **fully dynamically constructed computation graphs**
  - Not GAN (Generative Adversarial Network)!
- Supports
  - Training and evaluating models
  - Pre-trained analyze models (McParsey) for 40 language
    - Except Korean. (of course;)

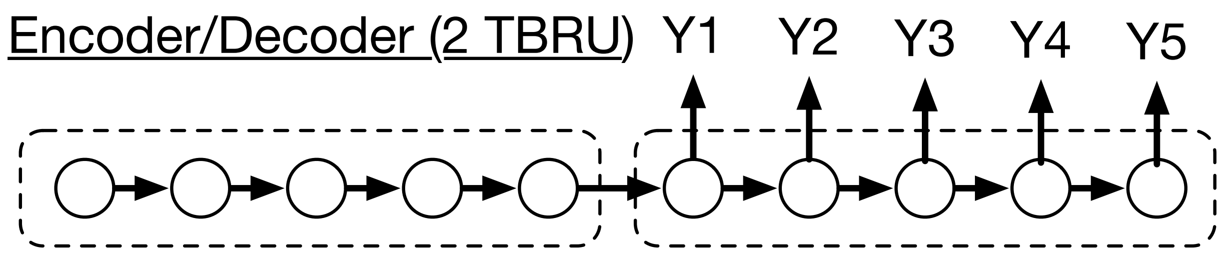
*Kong et al., (2017)

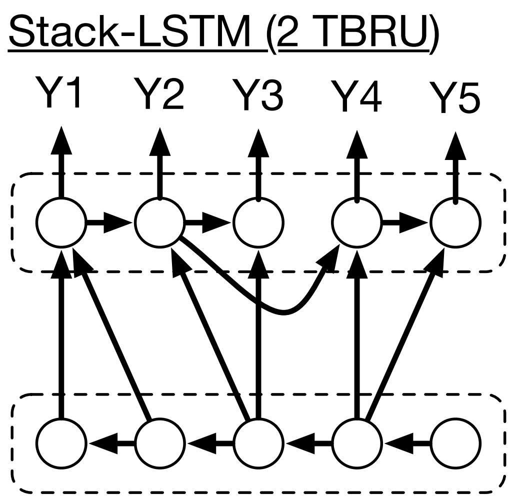
TBRU

- 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



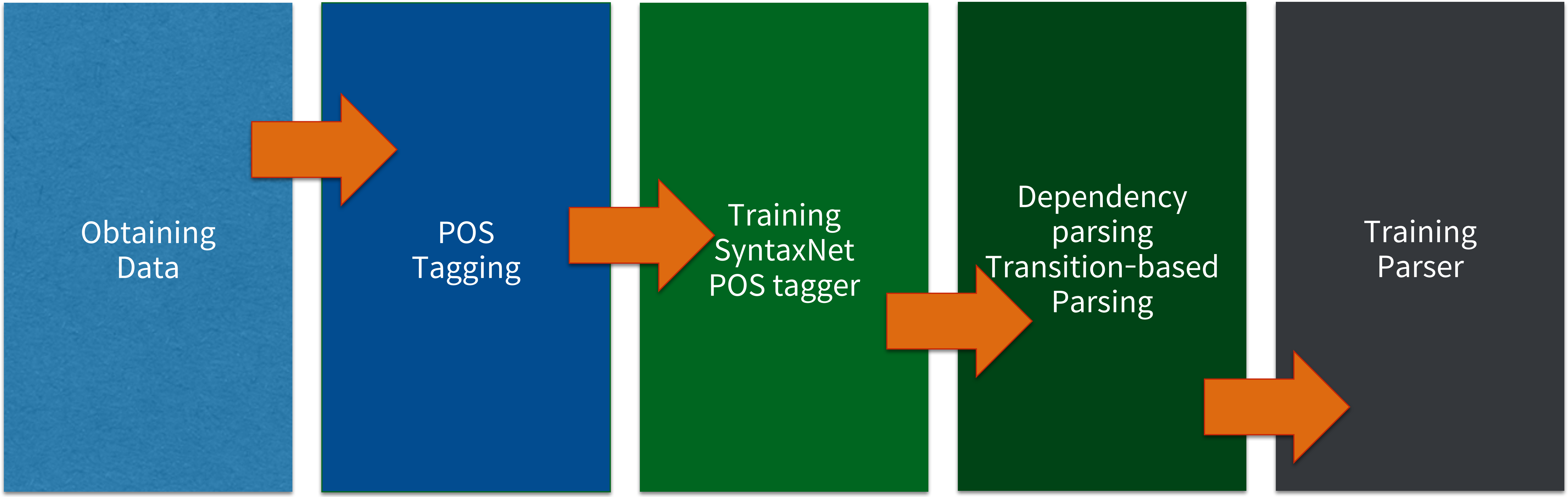






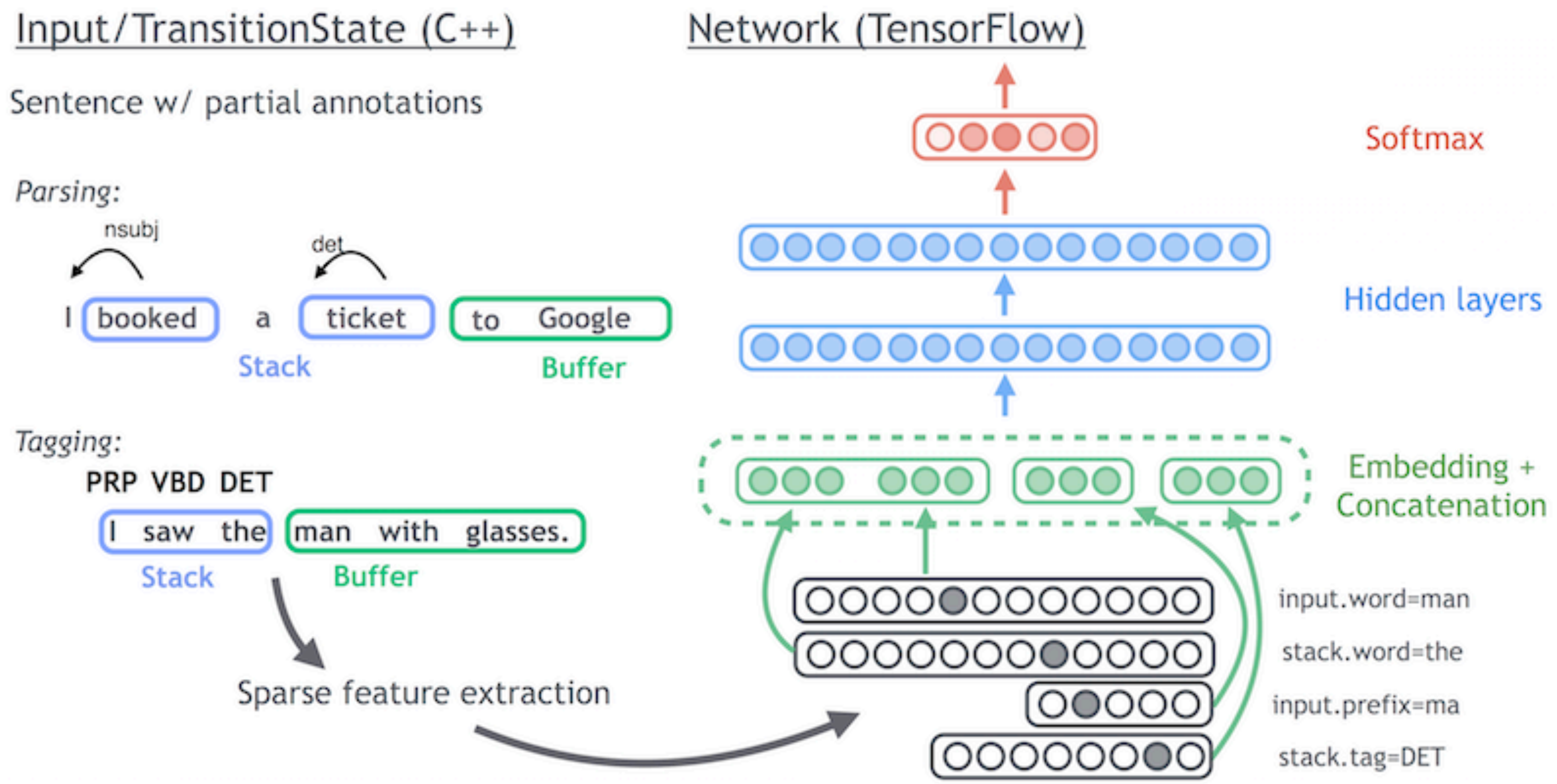
*Kong et al., (2017)

Generating NLP with SyntaxNet



SyntaxNet implementation

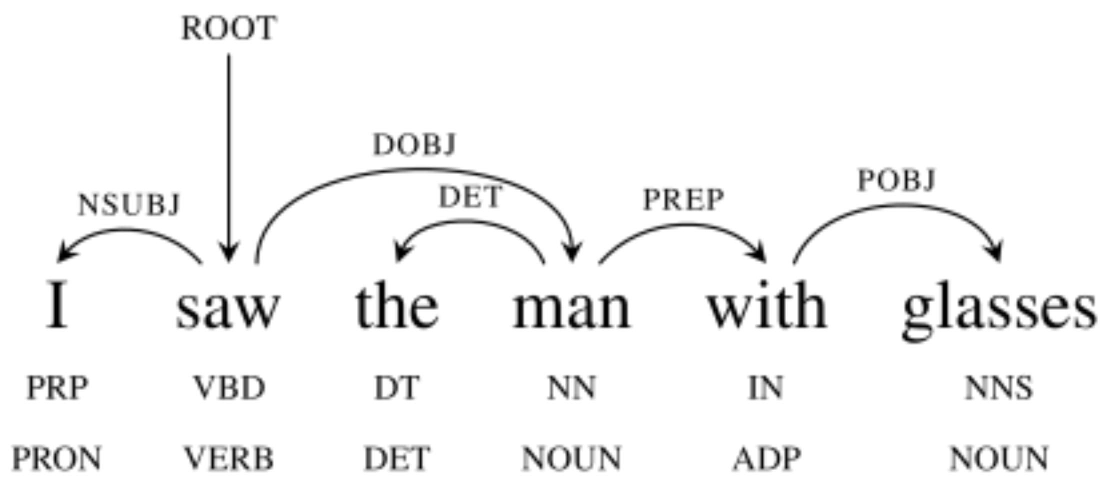
- Not a BOW (Bag-of-words) model
- Workflow
  - POS Tagging model
  - Preprocessing with tagger model
  - Dependency parsing

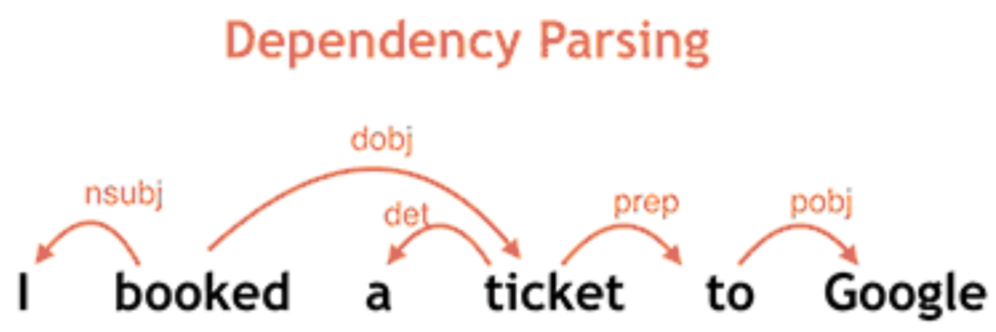


Feed-Forward SyntaxNet Architecture (Overview)

*github.com/tensorflow/tensorflow

SyntaxNet implementation



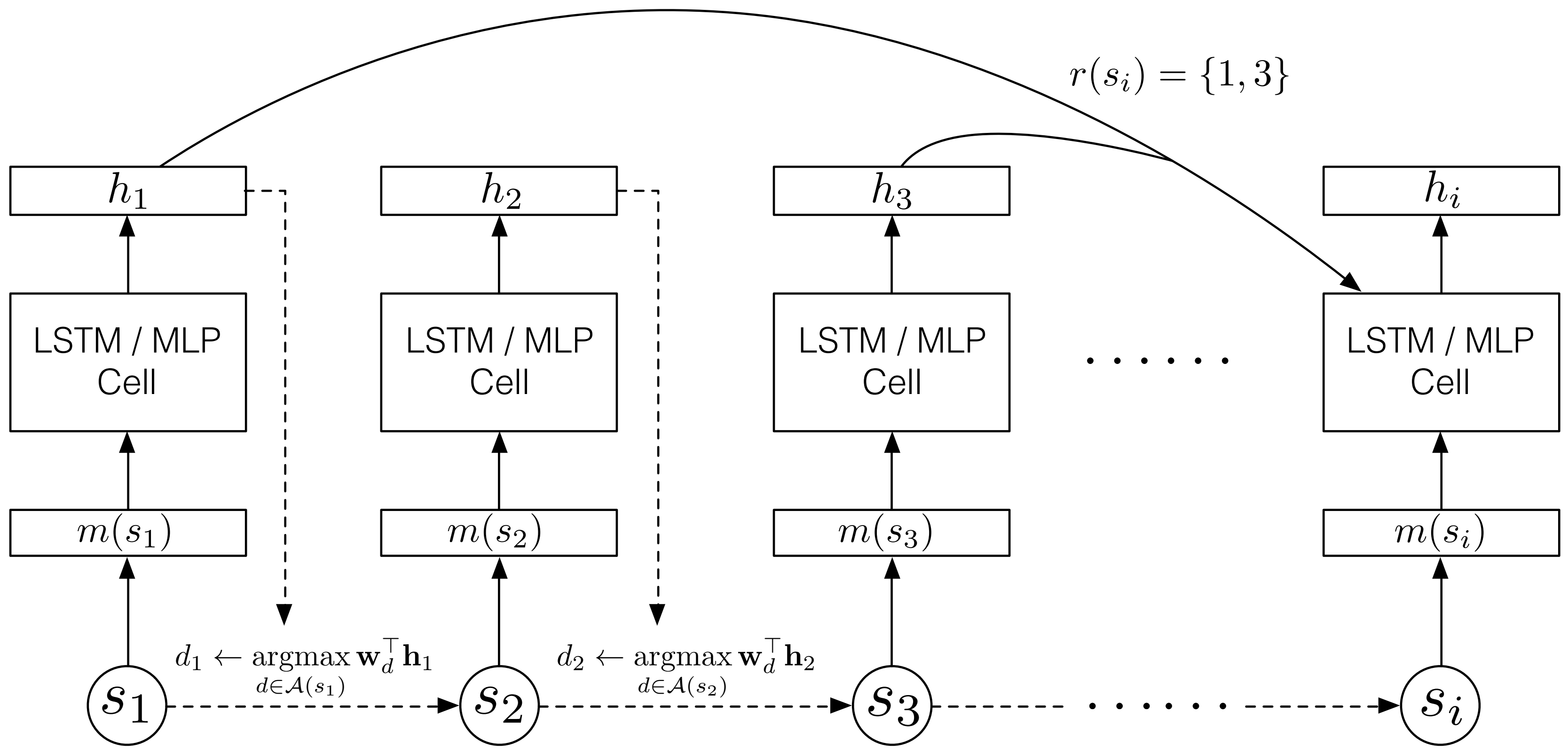


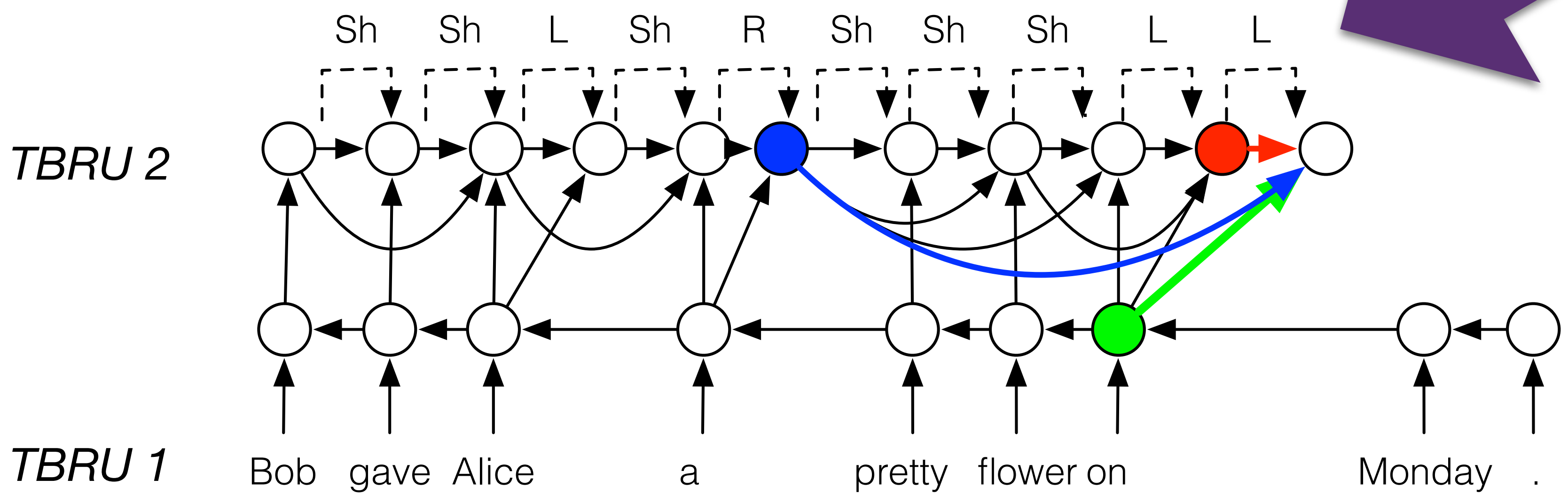
- Transition-based dependency parser
  - SHIFT, LEFT ARC, RIGHT ARC
- *"deviation"*
  - Configuration+Action
- Training
  - Local pre-training / global training

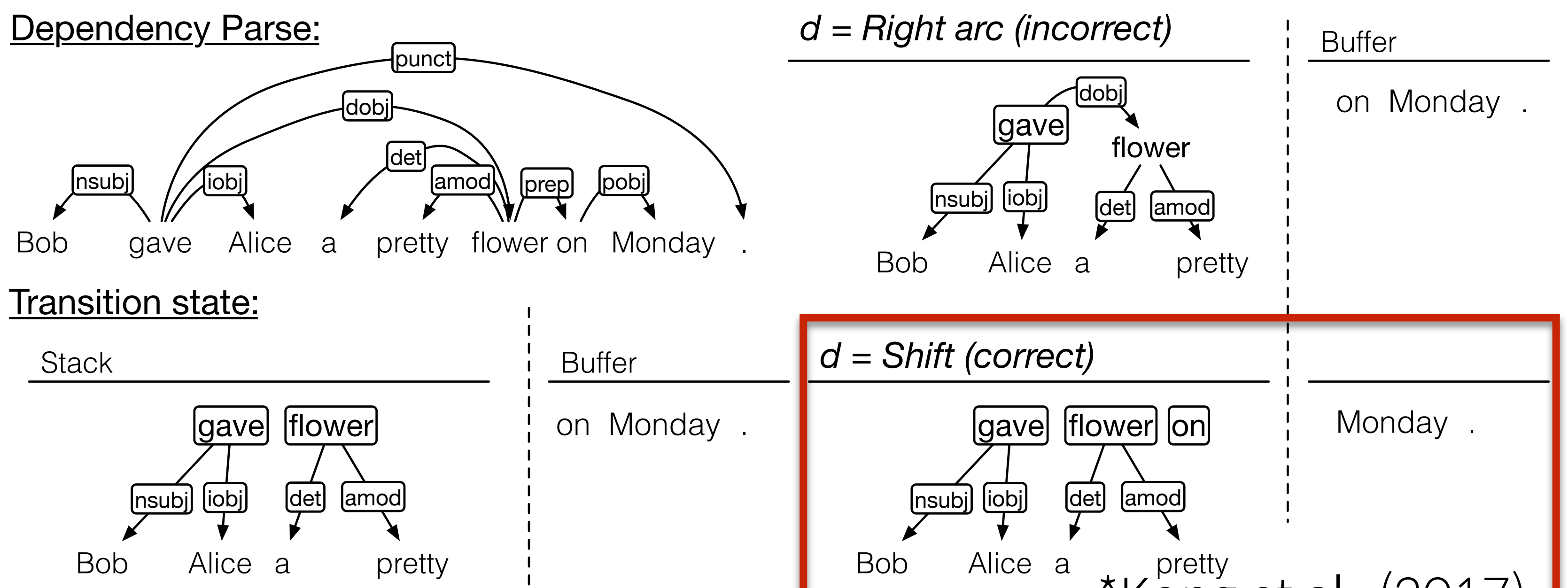
*Kong et al., (2017)

Dive into TBRU

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







*Kong et al., (2017)

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

[1] Kong et al., (2017)
[2] Andor et al., (2016)

Parsey McParseface

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

Model	News	Web	Questions
Ling et al. (2015)	97.44	94.03	96.18
Andor et al. (2016)*	97.77	94.80	96.86
Parsey McParseface	97.52	94.24	96.45

POS (part-of-speech) tagging

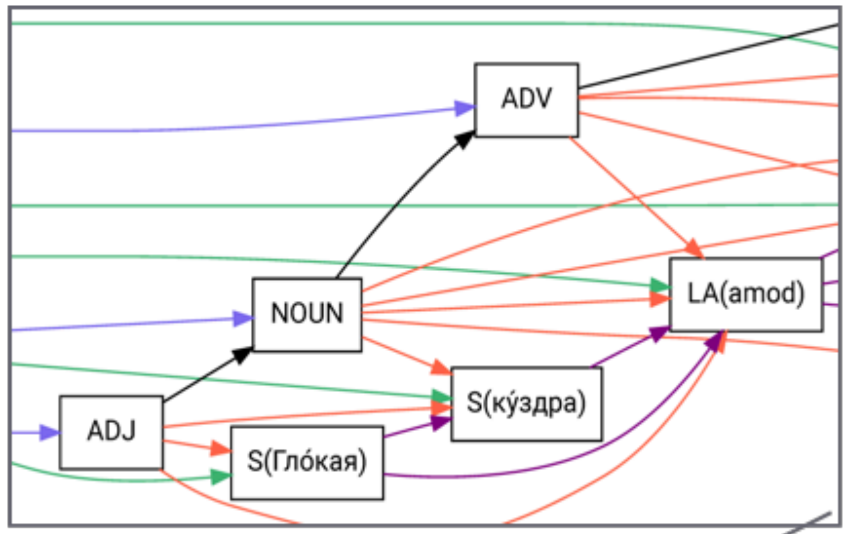
Model	News	Web	Questions
Martins et al. (2013)	93.10	88.23	94.21
Zhang and McDonald (2014)	93.32	88.65	93.37
Weiss et al. (2015)	93.91	89.29	94.17
Andor et al. (2016)*	94.44	90.17	95.40
Parsey McParseface	94.15	89.08	94.77

For different language domains

*github.com/tensorflow/tensorflow

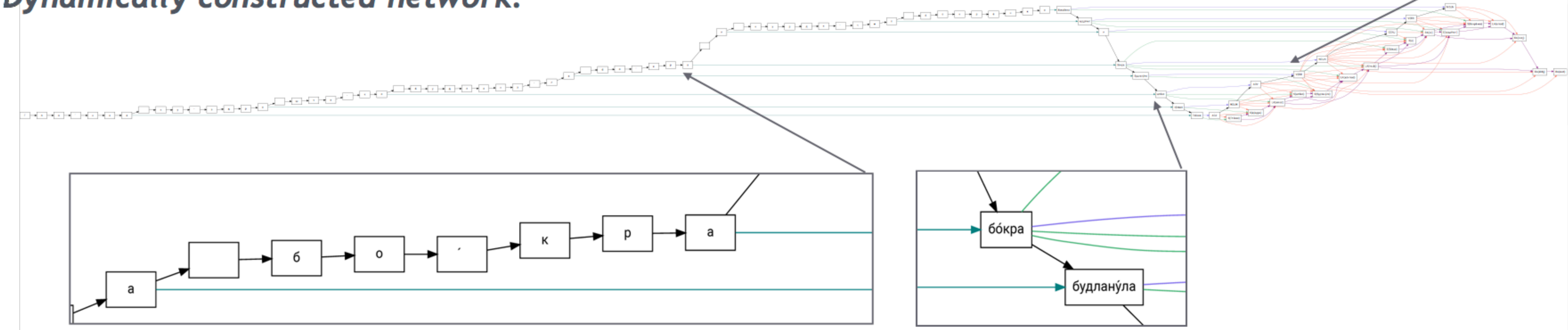
McParseface model / DRAGNN framework

ParseySaurus analysis:



POS and Syntax

Dynamically constructed network:

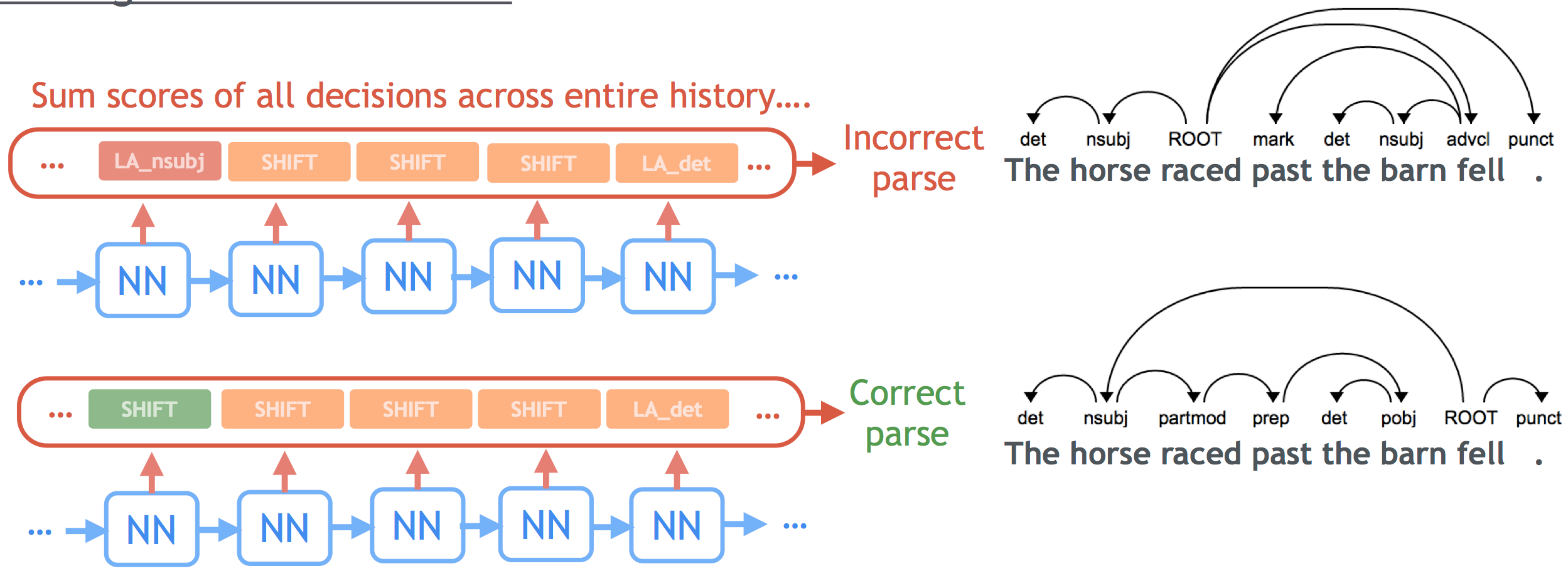


Character-based word representations

*github.com/tensorflow/tensorflow

SyntaxNet Architecture

Training with Beam Search:



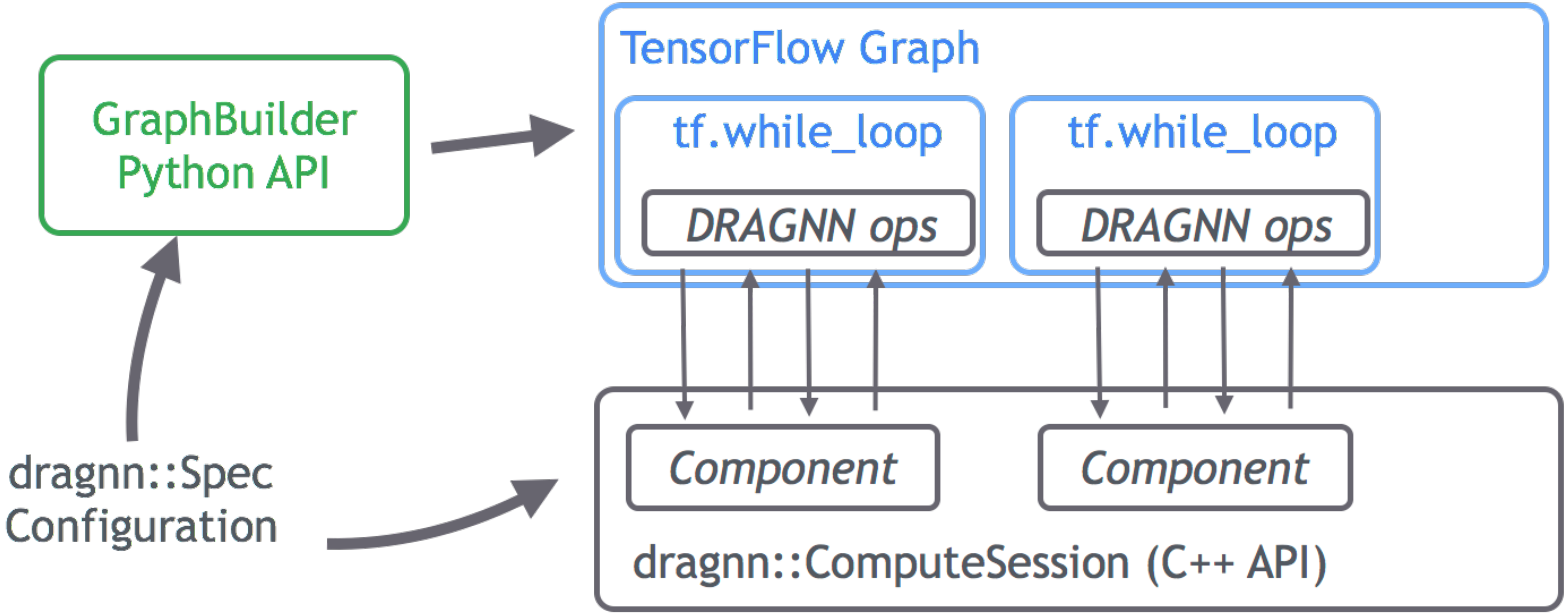
Update: maximize P(correct parse) relative to the set of alternatives

[Globally Normalized SyntaxNet Architecture (Overview)]

*github.com/tensorflow/tensorflow

DRAGNN implementation

- DRAGNN implementation on TensorFlow

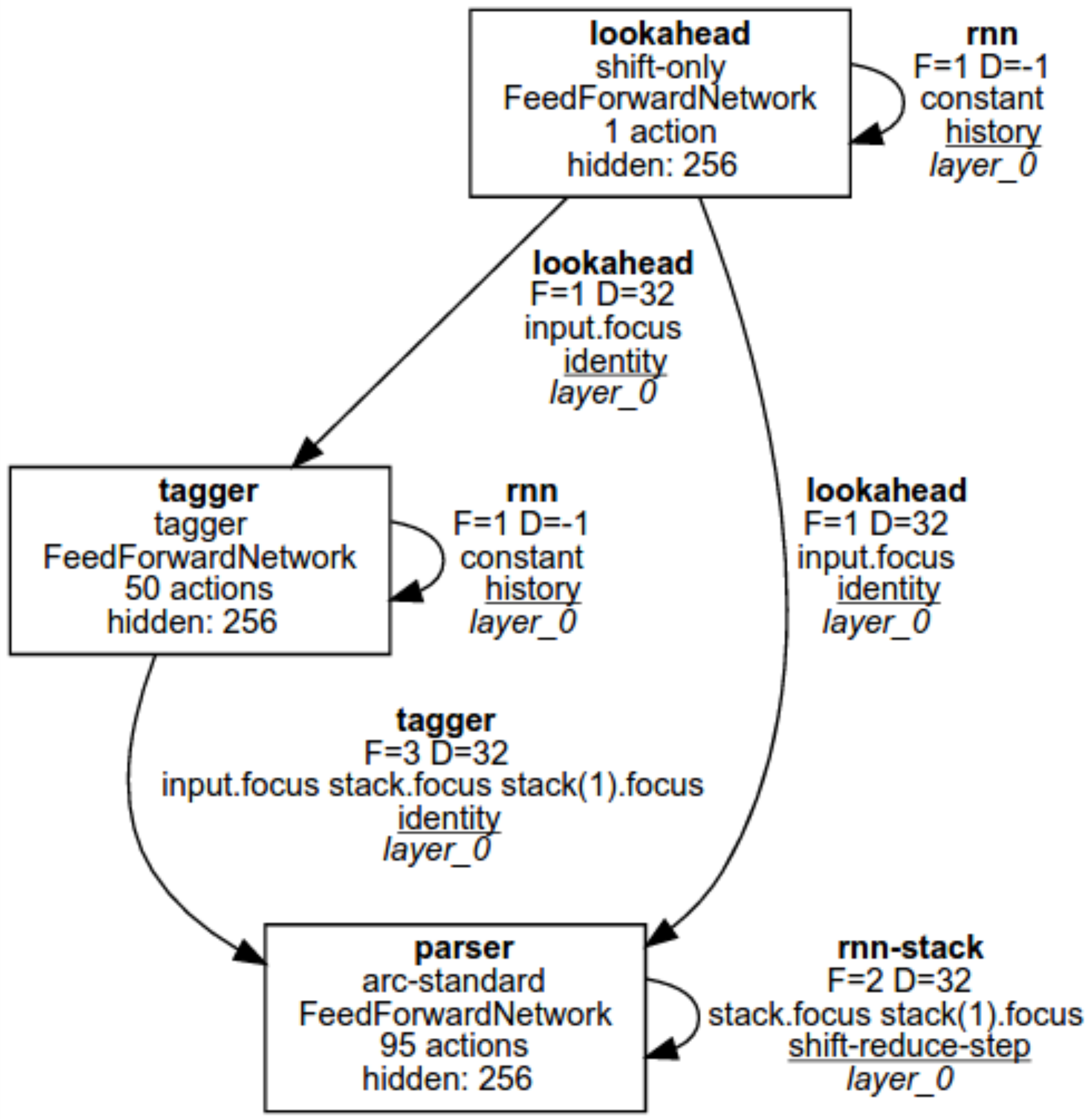


*github.com/tensorflow/tensorflow

Compute Graph

- Compute graph for SyntaxNet
  - Example case in TensorFlow repo.
- Three parts of NLP
  - Lookahead
  - Tagger
  - Parser
- Characteristics
  - Every model uses memory effect

```
master_spec = spec_pb2.MasterSpec()
master_spec.component.extend([lookahead.spec, tagger.spec, parser.spec])
HTML(render_spec_with_graphviz.master_spec_graph(master_spec))
```



*github.com/tensorflow/tensorflow

Why no Korean?

- Korean language-specific characteristics

- Solution?
  - Yes, I think. (testing now.)

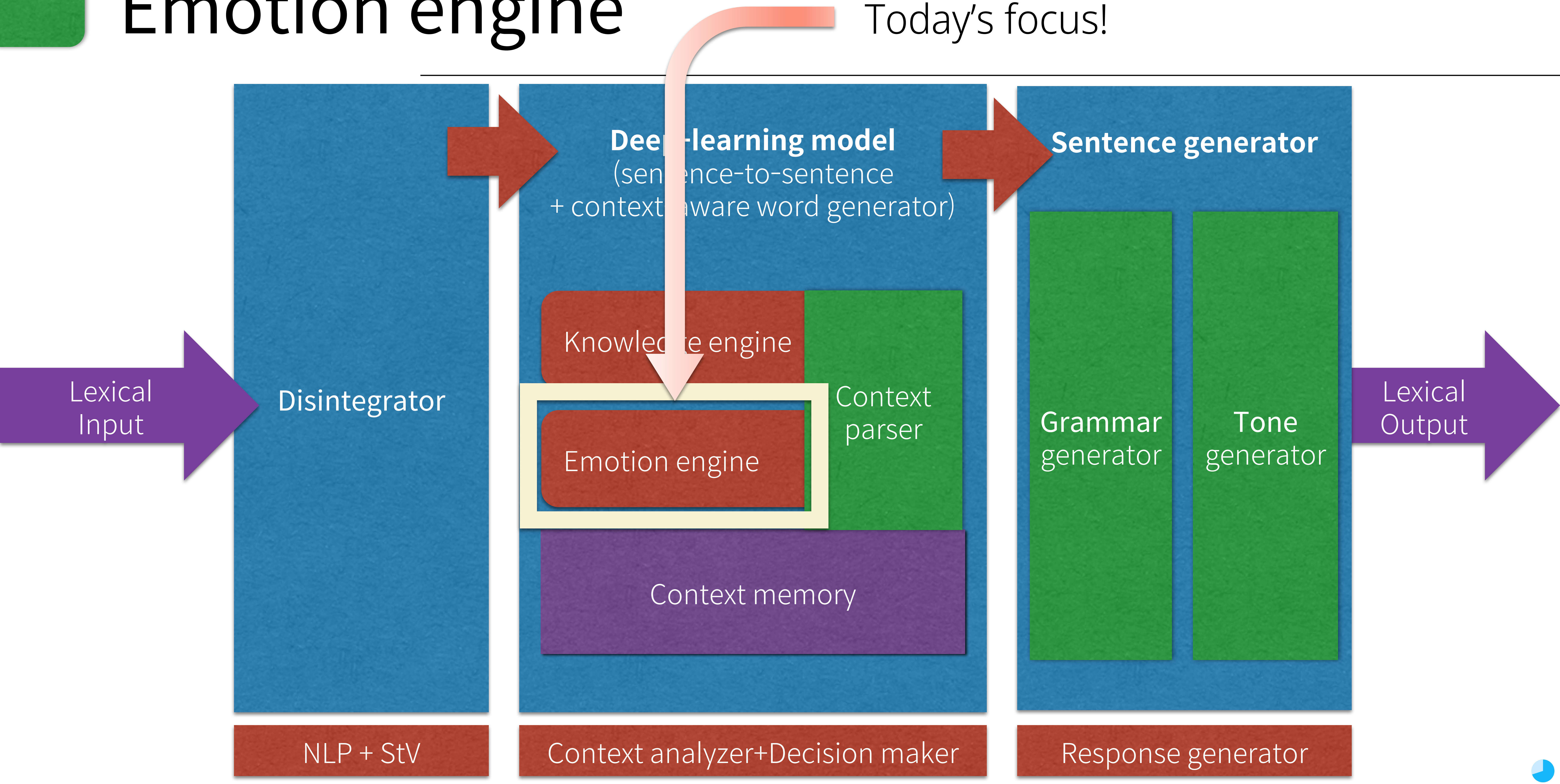
Now, let’s move to the emotion part.

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



Conversational context locator

- Using Skip-gram and bidirectional 1-gram distribution in recent text
- I ate miso soup this morning. => Disintegrate first

I	Eat	Miso soup	Today	Morning

  - **Bidirectional 1-gram set (reversible trigram)**: {(I,miso soup),Eat}, {(eat,today),miso soup}, {(miso soup,morning),today}

<I>	<EAT>	<FOOD>	Today	Morning

  - **Simplifying**: {(<I>,<FOOD>),<EAT>}, {(<EAT>,Today),<FOOD>}, {(<FOOD>,morning),Today}

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

  - **Distribution**: more simplification is needed
    - {(<I>,<FOOD>), <EAT>}, {(<TIME:DATE>,<EAT>), <FOOD>}, {(<FOOD>,<TIME:DAY>),< TIME:DATE>}
    - 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

나 | 오늘 | 아침 | 된장국 | 먹다

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

<I> | 오늘 | 아침 | <FOOD> | <EAT>

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

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

  - **Distribution**: more simplification is needed
    - {(<I>,<TIME:DAY>), <TIME:DATE>}, {(<TIME:DATE>,<FOOD>), <TIME:DAY>}, {(<TIME:DAY>,< EAT>),<FOOD>}
    - 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)

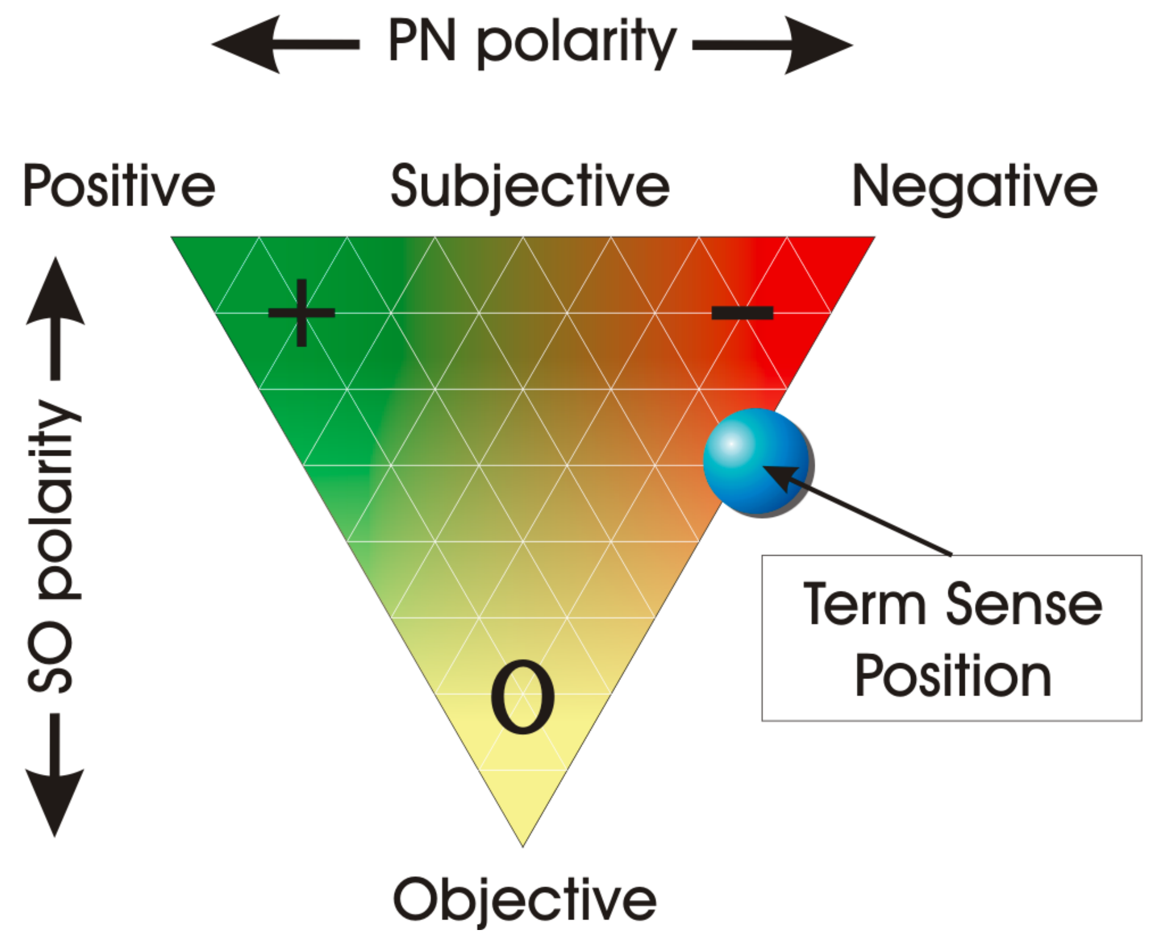
- Change from trigram to in-range tri pairs
- I ate miso soup this morning:

<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



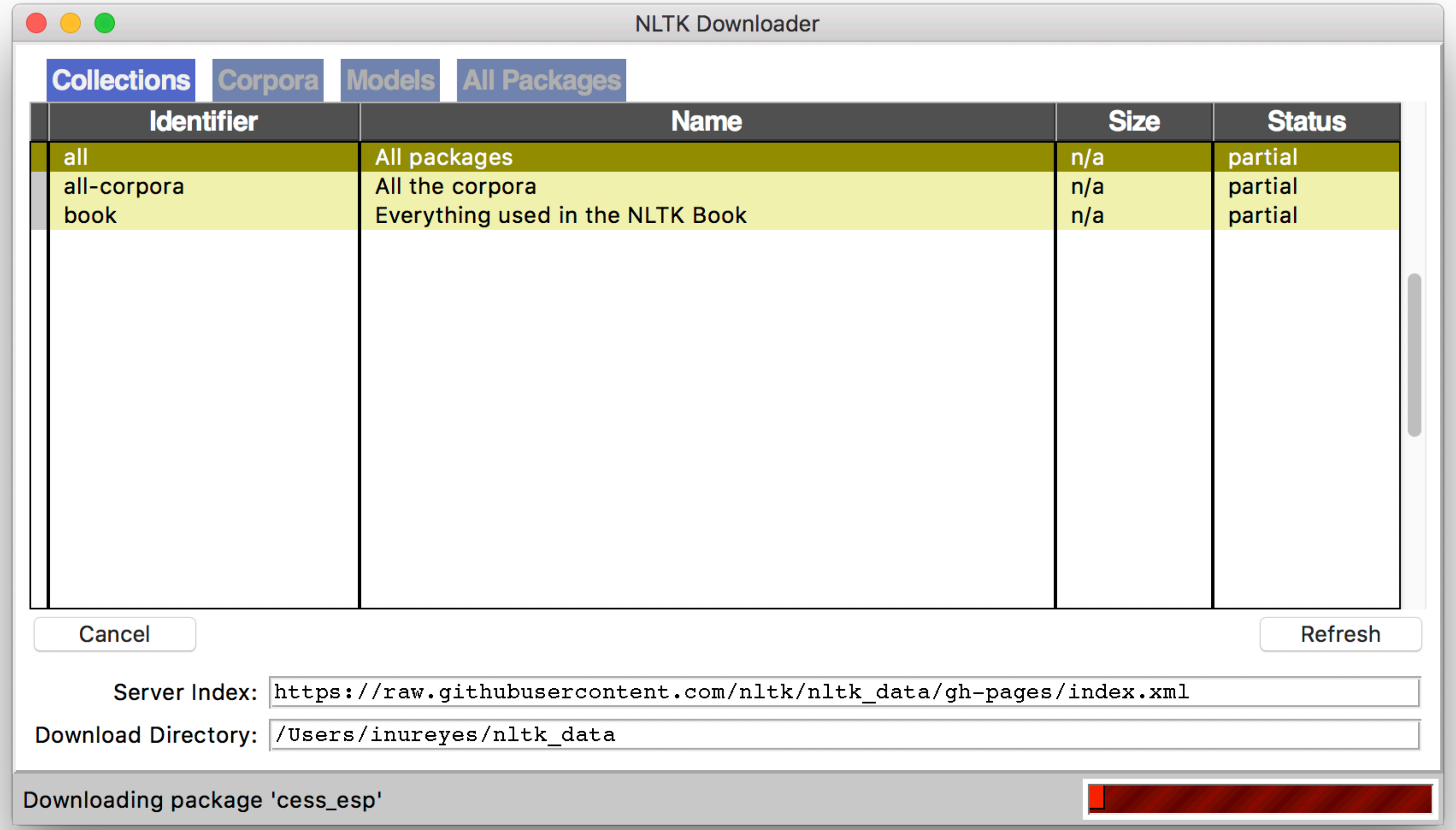
Position in senti-space:
[0.95, 0.05, 0.11, 0.89, 0.92, 0.08] → [1, 0, 0, 0, 0, 0] → 0x01

index: 1 2 3 4 5 6

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!



Downloading NLTK dataset

```python
import nltk
nltk.download()
```

Making emotional context locator

- Get emotional flag from sentence

Sample test routine for Sentimental state

```
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
```

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

Adj. only

```
{'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

```
{'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

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

```
<maimed.s.01: PosScore=0.0 NegScore=0.0>
<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>
<unfit.a.01: PosScore=0.25 NegScore=0.0>
```

2. Translate words into Korean

```
<불구의.s.01: PosScore=0.0 NegScore=0.0>
<알맞다.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>
```

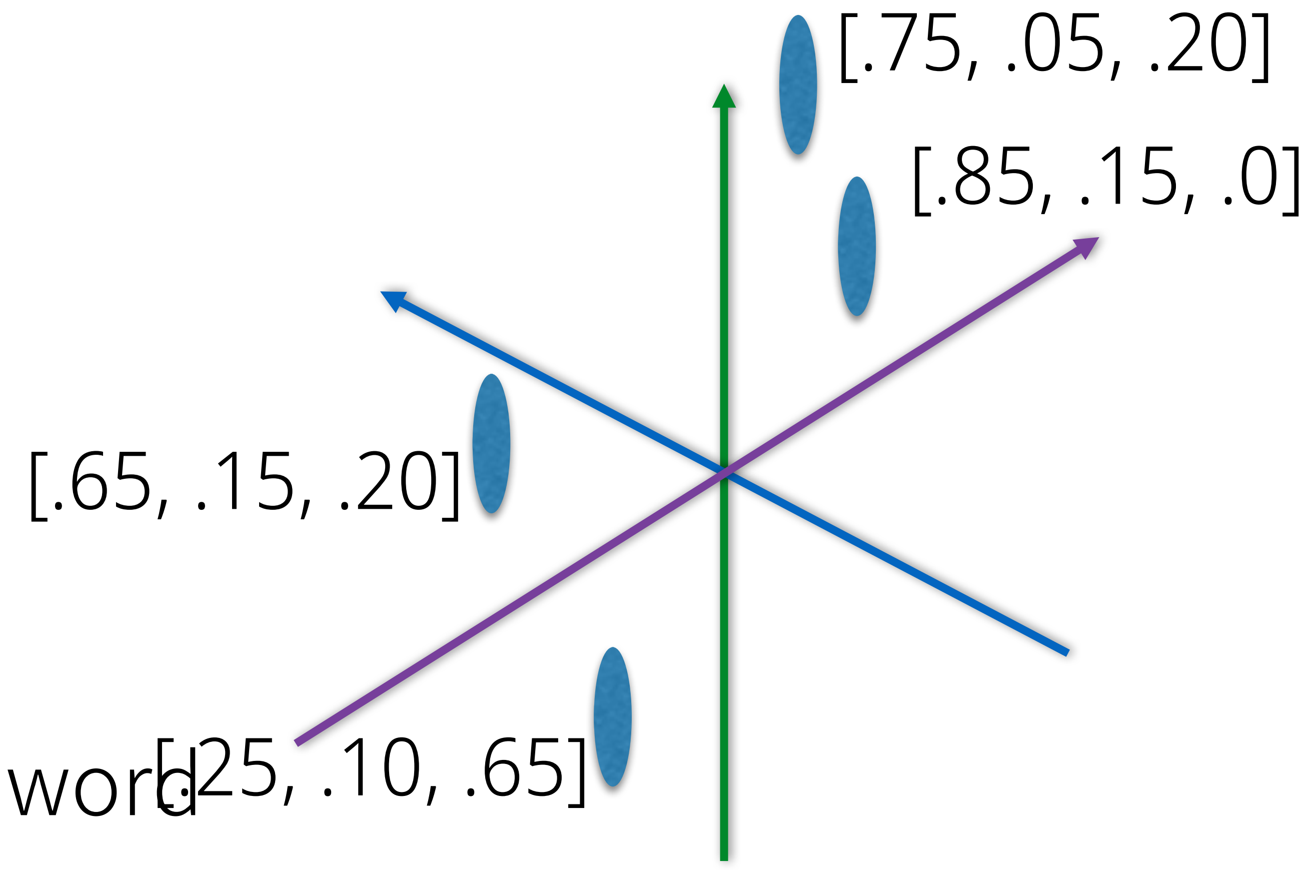
3. Treat synonym

4. Choose the score from 'representative word'

```
<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



WordVec Space

*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 BETA

- 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