Building AI **Chat bot** using **Python 3** & **TensorFlow**

Jeongkyu Shin
Lablup Inc.

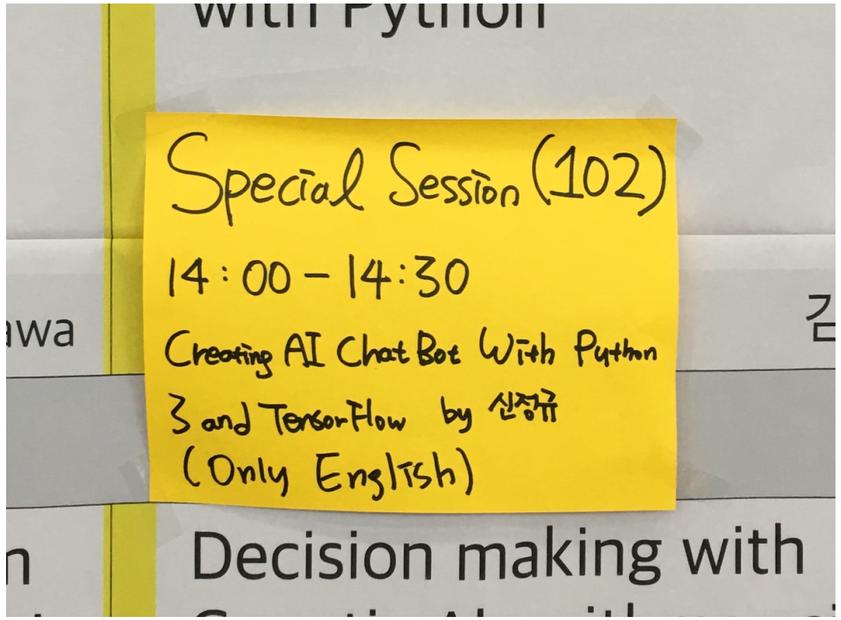
I'm

신정규 / Jeongkyu Shin / @inureyes

- Humble business man
  - **Lablup** Inc. (***All members are speaking*** *in PYCON APAC 2016!*)
- Open-source devotee
  - **Textcube** maintainer
  - Play with some (open||hidden) projects / companies
- Physicist / Neuroscientist
  - Studied *information processing procedure* in *Brain*
  - **Ph.D in Statistical Physics** (complex system)
  - Major in **Physics** / **Computer science**

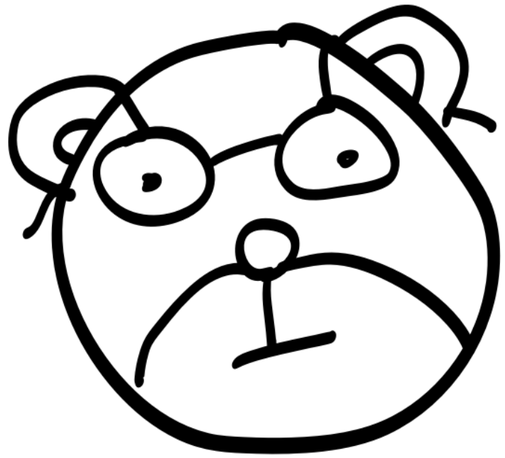
> runme –-loop=2

- Became the first man to get 2 official presenter shirts in PyCON APAC 2016!
- *Are you ready? (I'm not ready)



*Parody of something. Never mind.

Welcome to my garage!



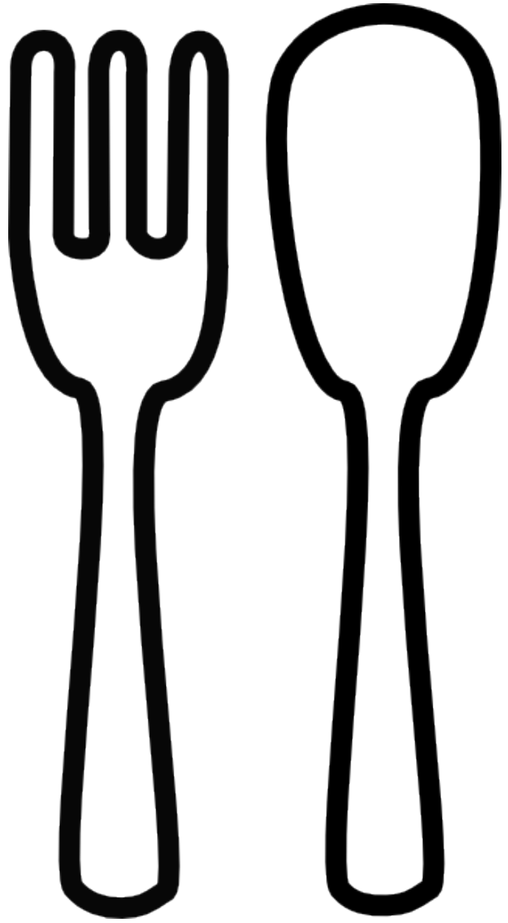
Tons of garbage here!

“First with the head, then with the heart.”

–Bryce Courtenay's *The Power of One*

Today’s Entree: Chat bot

- Python 3
  - Twitter Korean Analyzer / Komoran with KoNLPy / pandas
- TensorFlow
  - 0.8 -> 0.9 -> 0.10RC
- And special sauce!
  - Special data with unique order
  - Special python program to organize / use the data!



Ingredients for today's recipe

- Data
  - Test: **FAS** dataset (26GB)
  - **Today:** “*Idolm@ster*” series and etc.
- Tools
  - TensorFlow + Python 3
- Today’s insight
  - Multi-modal Learning models and model chaining

I'm **not sure** but
I'll **try to explain** the
whole process I did

Game screenshot* (c) CAVE
Forkcrane* (c) Iconix

And I assume that you **already** have **experience** / knowledge about **machine learning** and **TensorFlow**

Illustration *(c) marioandluigi97.deviantart.com

Things that will **not** be covered today

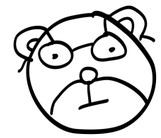
- Phase space / embedding dimension
- Recurrent Neural Network (**RNN**)
- GRU cell / LSTM cell
- Multi-layer stacking
- Batch process for training
- Vector representation of language sentence
- Sequence-to-sequence model
- Word2Vec / Senti-Word-Net

One day in **Seoul** Itaewon, **2013**

All started with dinner talks of neuroscientists…

“When will AI-based program pass Turing test?”

“I believe it will happen before 2020.”



“Is it too fast to be true?”

“Weak intelligence will achieve the goal with accelerated technology advances and our understanding about human brain.”

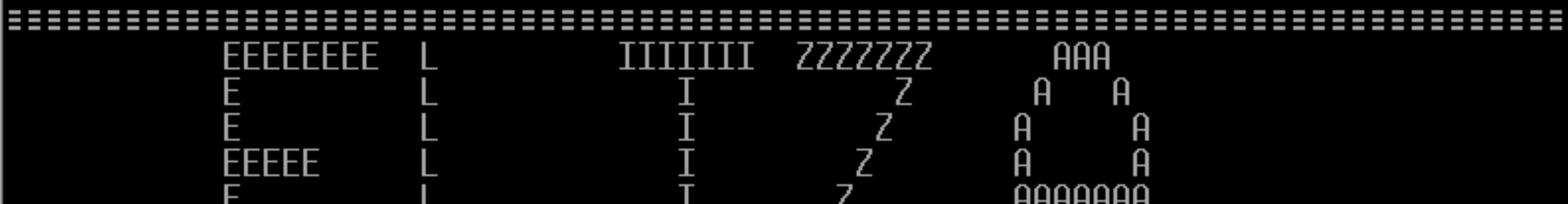
“Do you really believe that it will happen in that short time?”

“Ok, then, let’s make a small bet.”

...and I started making my own chat bot from next month of the day.

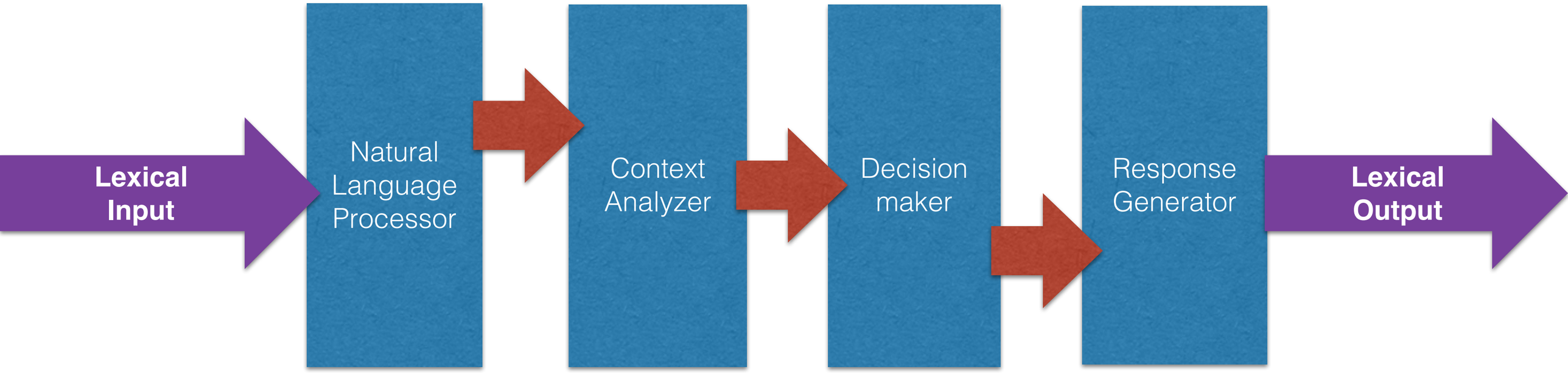
What is chat bot?

- “Chatting bots”
- One of the
  - *Oldest* Human-Computer Interface (**HCI**) based machines
  - Challenging lexical topics
- Interface: **Text** → Speech (**vocal**) → Brain-Computer Interface (**BCI**)
- Commercial UI: ***Messengers!***

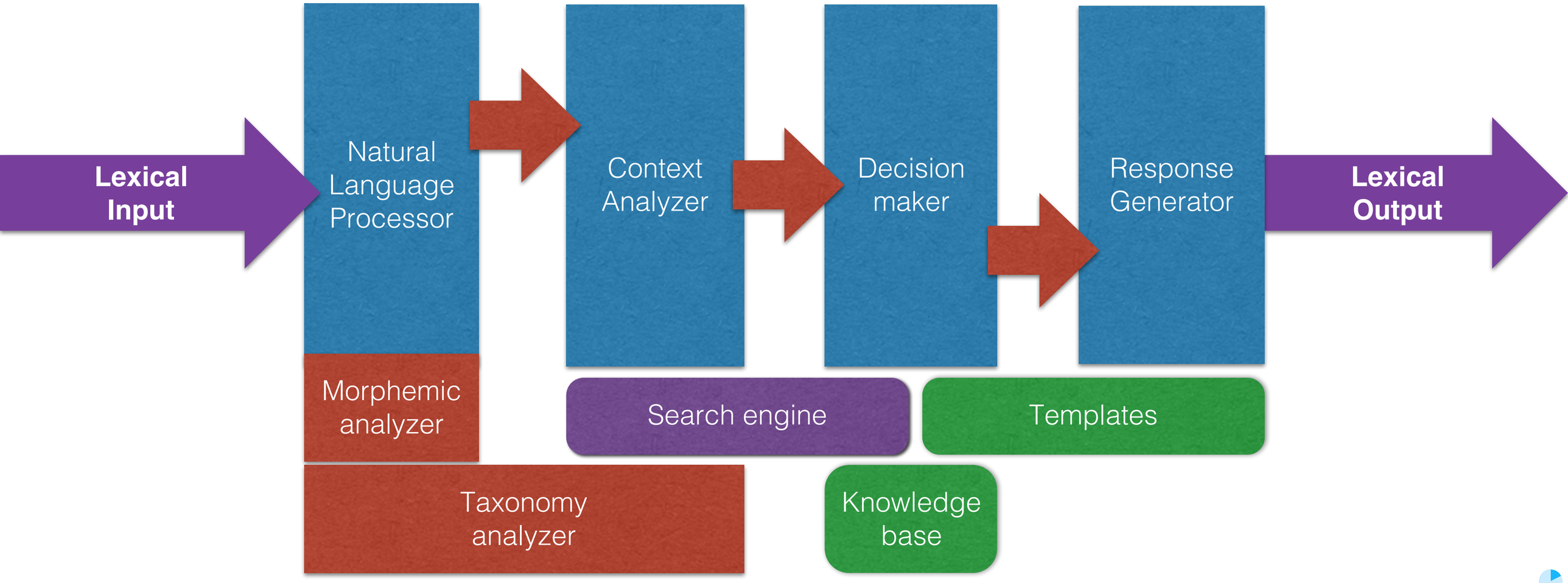



```
ELIZA > (PLEASE TYPE IN ALL CAPS) WHAT'S YOUR NAME DEAR ?
AMIT
ELIZA > HI, I'M ELIZA. WHAT DO YOU WANT TO TALK ABOUT ?
AMIT > CAN I TALK ABOUT MY PROBLEM ?
ELIZA > SURE... !
AMIT > DUE TO ON GOING EXAMS IN COLLEGE, I HAVE A LOT OF STRESS.
ELIZA > PLEASE RELAX A LITTLE. SLEEP WELL.
AMIT > THANKS FOR YOUR ADVICE.
ELIZA > NO MENTION.
AMIT > BYE
ELIZA > BYE AND KEEP IN TOUCH...
```

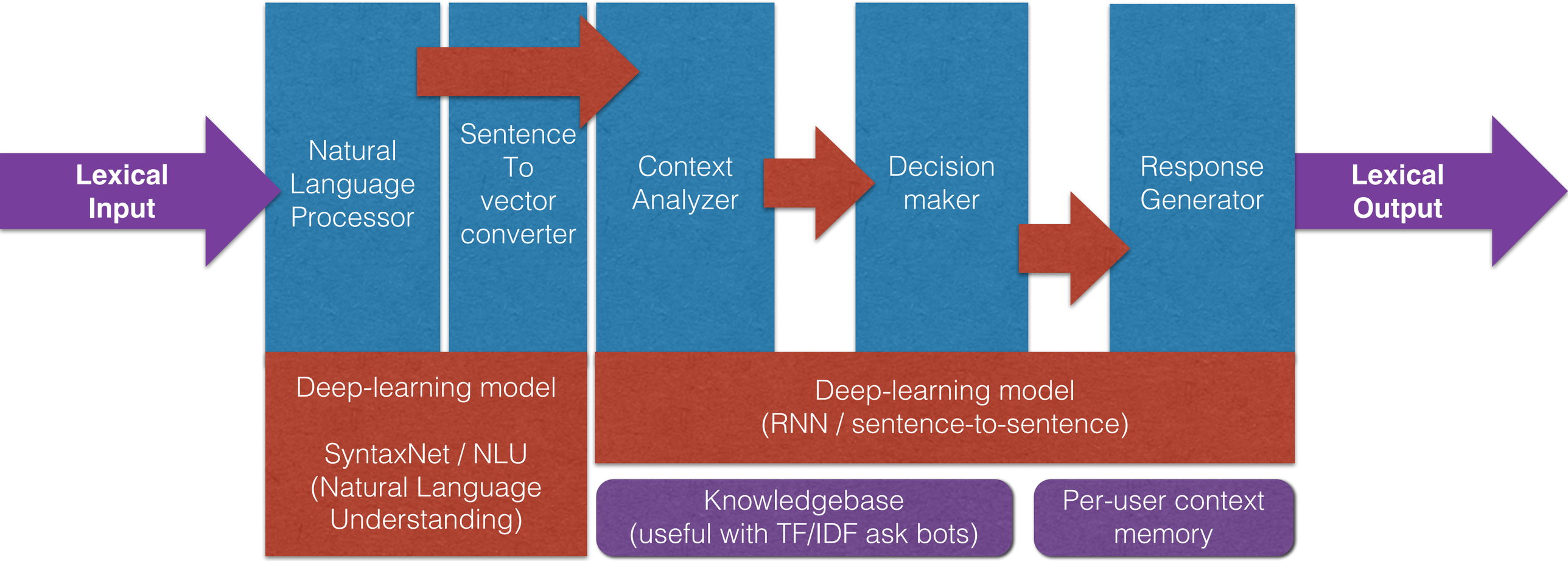
Basic chat bot components



Traditional chat bots



Chat-bots with Machine Learning



Problems

- 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
- Korean-specific problems
  - Dynamic type-changes
  - Postpositions / conjunction (Josa hell)

헬조사
Hell Josa

=

The great wall
of Korean ML+NLP
Like
ActiveX+N*+F*
In Korean Web

We **expect** these but…

Clip art *Lego ©

We **got** these.

…How can I assemble them?

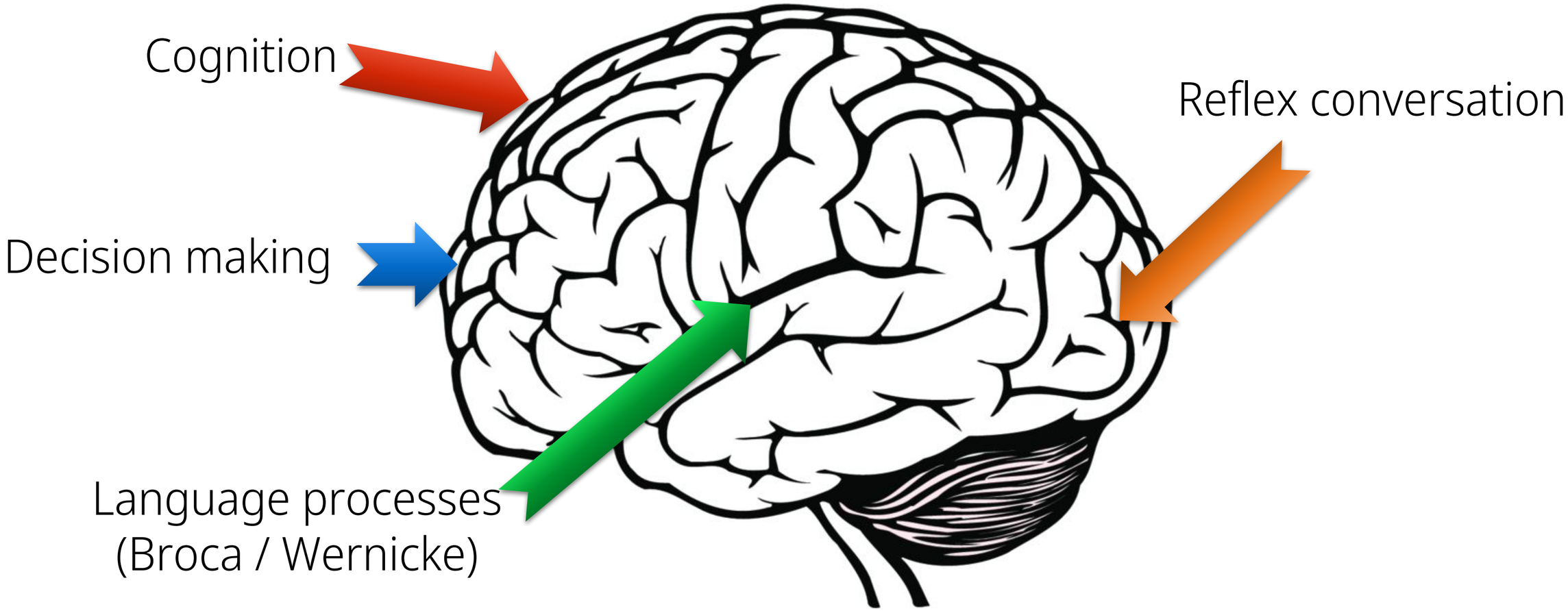
Photo * © amazon.com

Back to the origin

What I learned for 9 years…

How brain works

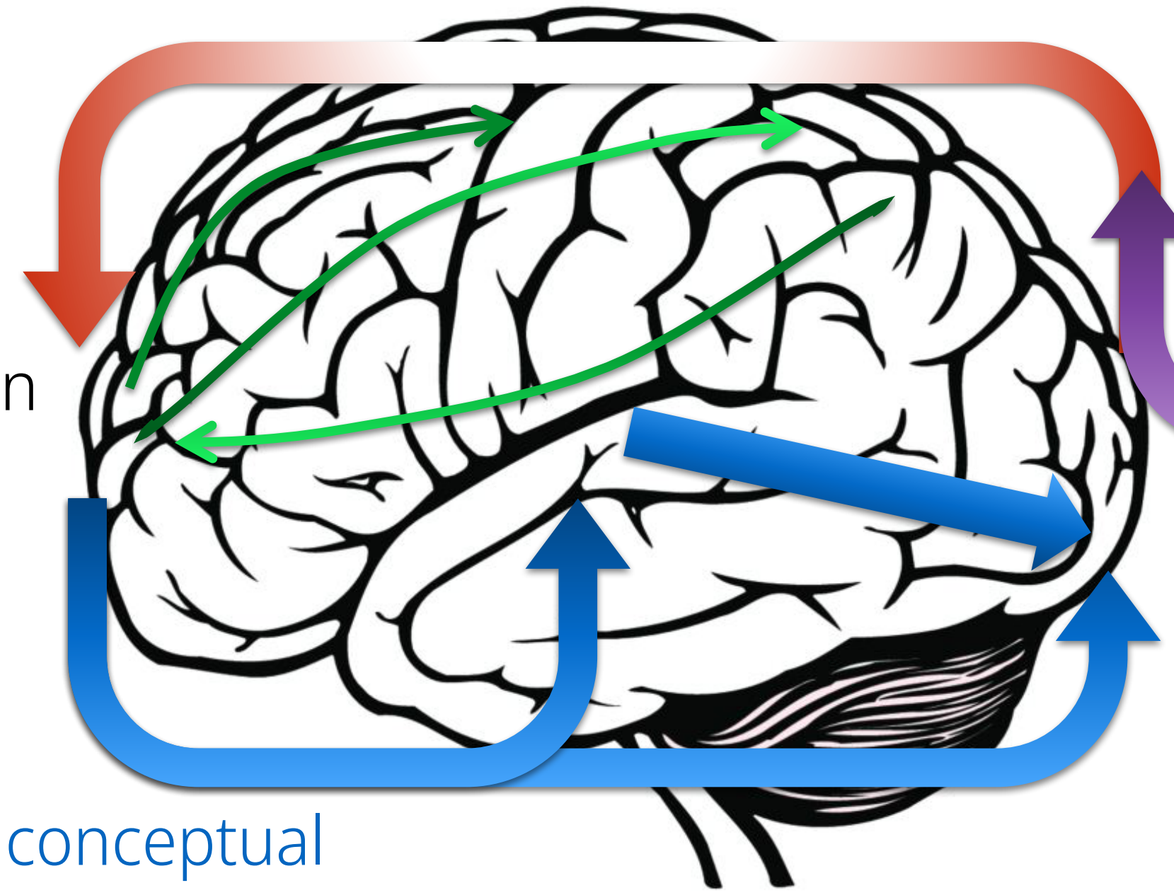
- Parallelism: performing a task at separated areas



Information pathway during conversation

- During conversation:

2. Send information



1. Preprocessing

3. Context recognition

4. Spread / gather processes to determine answer

6. Postprocessing to generate sentence

5. Send conceptual response to parietal lobe

Understanding brain process

- Intelligence / cognitive tasks
  - Temporal information circuit between prefrontal-frontal lobe
- Language processing
  - Happens after backward information signal
  - Related to somatosensory cortex activity
- Ok, ok, then?
  - Language process is highly separated in brain
  - Integration / disintegration process is very important

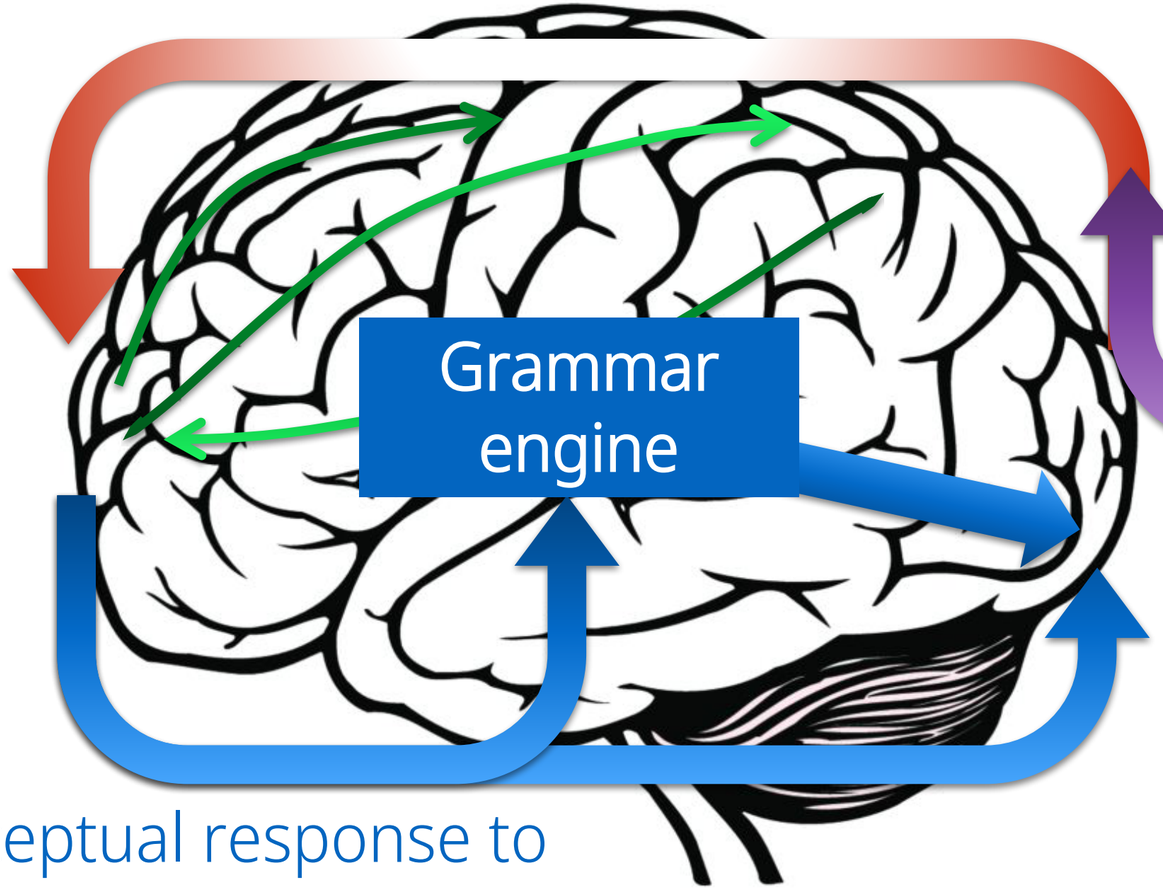
Architecturing

- Separate the dots
  - Simplifying information to context analyzer
  - Generates complex response using diverse models
- Sentence generator
  - **Grammar** generator model
    - Simple word sequence to be complete sentence
  - **Tone** generator model
    - Change sentence sequence tones with specific tone

Ideas from structure

- During conversation:

2. Send information



1. Disintegrator

3. Context parser

4. Decision maker using ML model

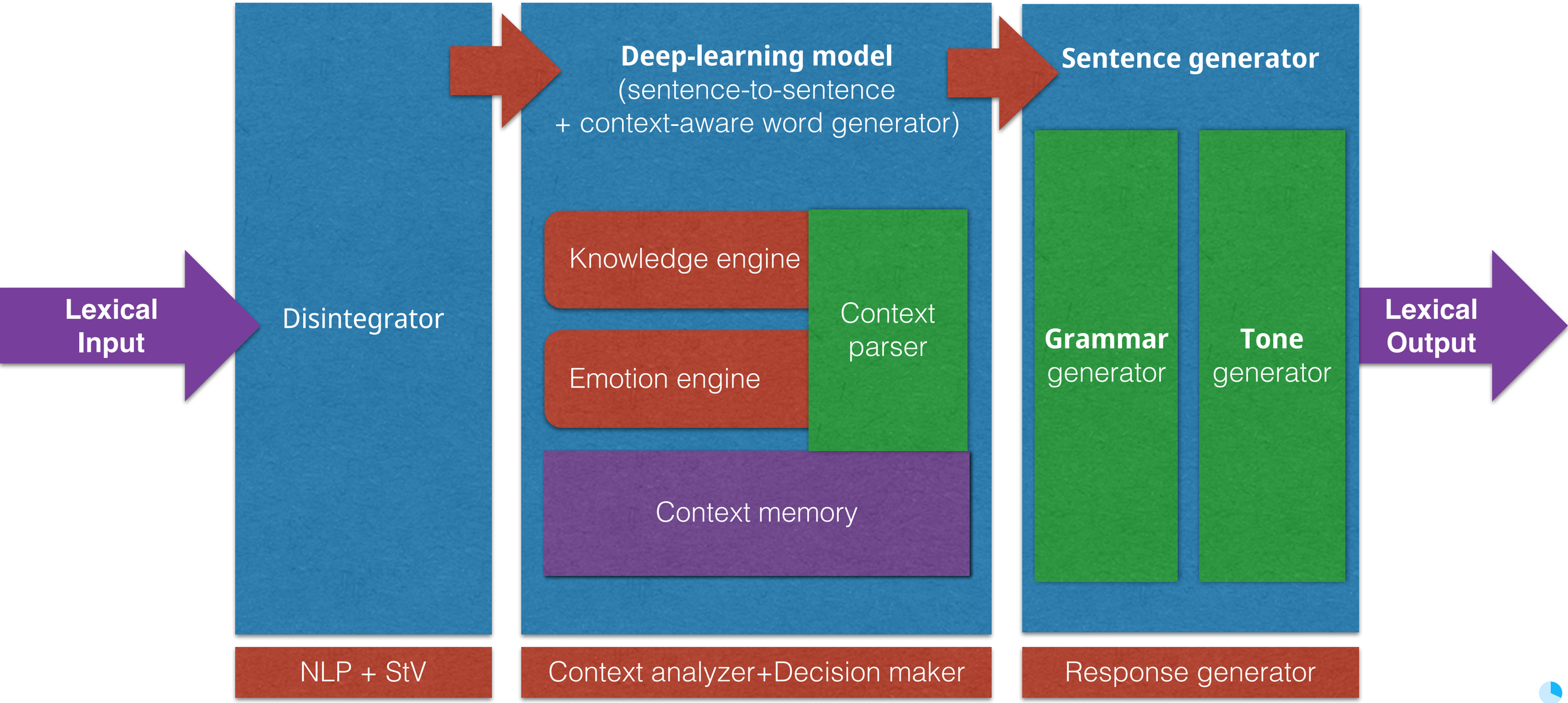
6. Postprocessing with **tone engine** to generate sentence

5. Send conceptual response to **Sentence generators**

Ideas from structure

- Multi-modal model
  - Disintegrator (to simplify sentence into morphemes)
  - Bot engine
    - Generates morpheme sequence
  - Grammar model
    - Make meaningful sentence from morpheme sequence
  - Tone model
    - Change some conjunction (*eomi*) / words of grammar model result

Final structure



Making models

The importance of ***Proto***typing

Creating ML models

Prepare

train **dataset**
test dataset
Runtime environment

Define

input function
step function
evaluator
batch

Make

Estimator
Optimizer

Do

Training
Testing
Predicting

Creating ML models

Prepare

train **dataset**
test dataset
Runtime environment

Define

input function
step function
evaluator
batch

Make

Estimator
Optimizer

Do

Training
Testing
Predicting

Creating ML models

Prepare

train **dataset**
test dataset
Runtime environment

Define

input function
step function
evaluator
batch

Make

Estimator
Optimizer

Do

Training
Testing
Predicting

Creating ML models

Prepare

train **dataset**
test dataset
Runtime environment

Define

input function
step function
evaluator
batch

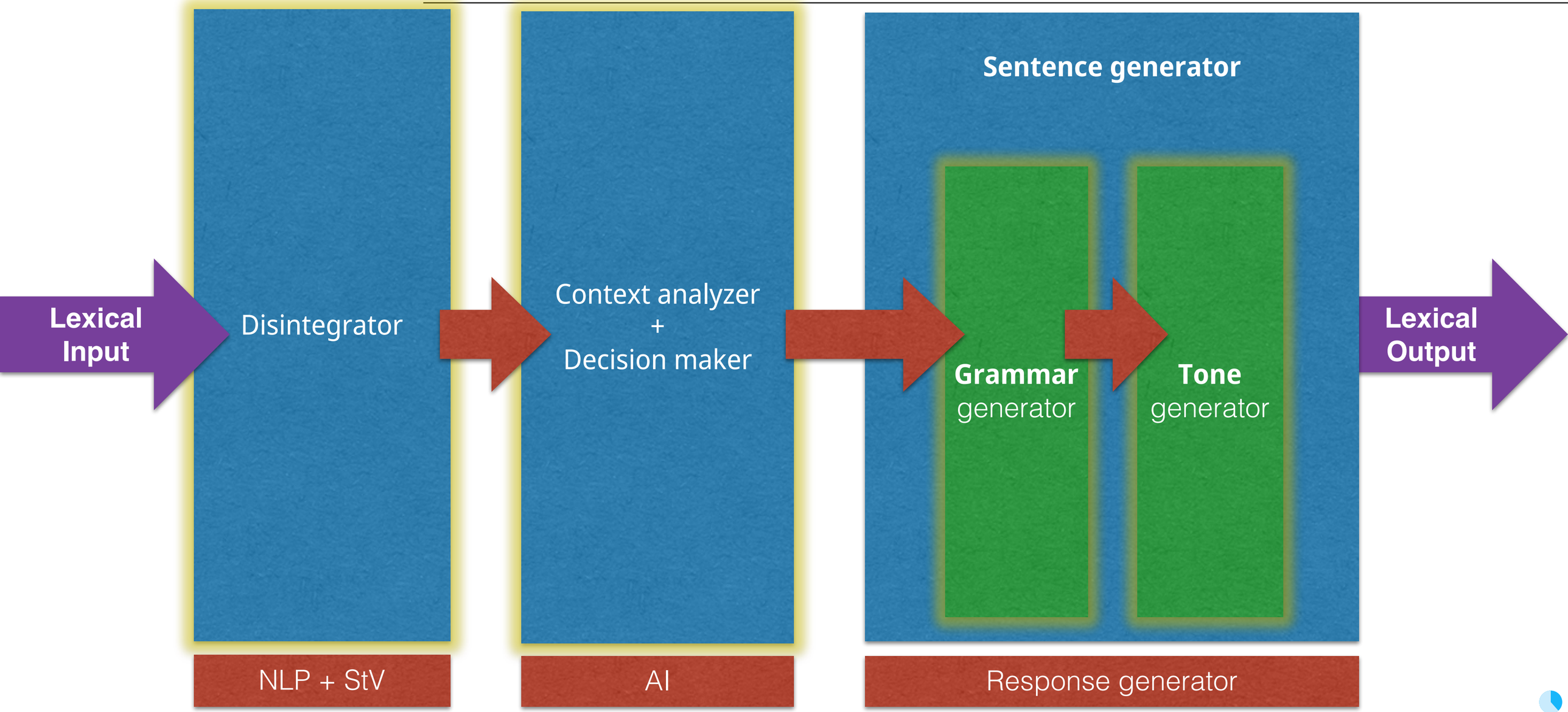
Make

Estimator
Optimizer

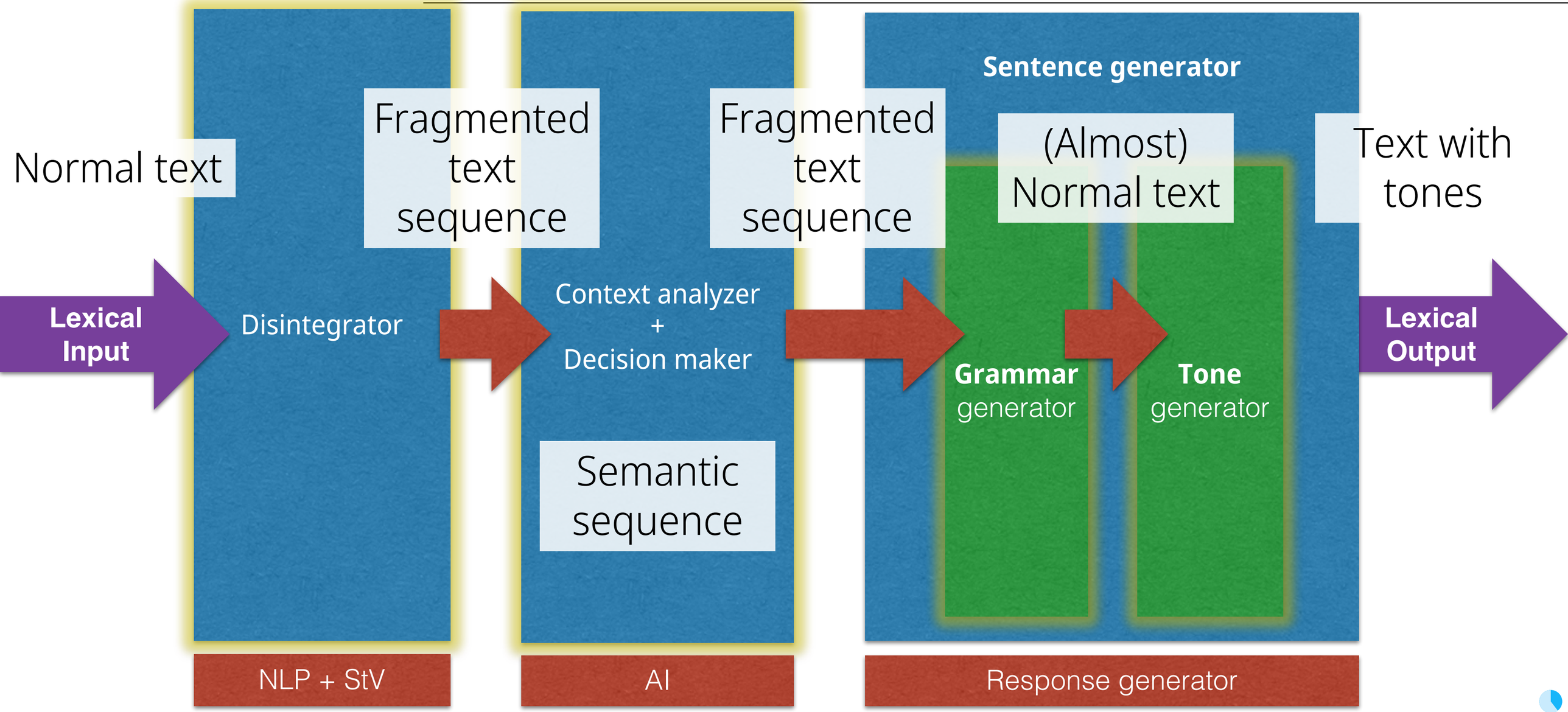
Do

Training
Testing
Predicting

Model chain order



Model chain order



Disintegrator

- a.k.a. morpheme analyzer for speech / talk analysis
- Input
  - Text as conversation
- Output
  - Ordered word fragments

Disintegrator

- *Twitter* Korean analyzer
  - Compact and very fast
  - Can be easily adopted with KoNLP package
  - Komoran can be a good alternative (with enough time)
- *Komoran* with ko_restoration package (https://github.com/lynn-hong/ko_restoration)
  - Increases both model training accuracy / speed
  - However, it is soooooooo slow… (> 100 times longer execution time)

Disintegrator

get_graining_data_by_disintegration

```
def get_training_data_by_disintegration(sentence):
    disintegrated_sentence = konlpy.tag.Twitter().pos(sentence, norm=True, stem=True)
    original_sentence = konlpy.tag.Twitter().pos(sentence)
    inputData = []
    outputData = []
    is_asking = False

    for w, t in disintegrated_sentence:
        if t not in ['Eomi', 'Josa', 'Number', 'KoreanParticle', 'Punctuation']:
            inputData.append(w)
    for w, t in original_sentence:
        if t not in ['Number', 'Punctuation']:
            outputData.append(w)
    if original_sentence[-1][1] == 'Punctuation' and original_sentence[-1][0] == "?":
        if len(inputData) != 0 and len(outputData) != 0:
            is_asking = True # To extract ask-response raw data
    return ' '.join(inputData), ' '.join(outputData), is_asking
```

Sample disintegrator

- Super simple disintegrator using twitter Korean analyzer (with KoNLPy interface)

```
(venv) disintegrator » python test.py
Original                                : 나는 오늘 아침에 된장국을 먹었습니다.
Disintegrated for bot / grammar input : 나 오늘 아침 된장국 먹다
Training data for grammar model output: 나 는 오늘 아침 에 된장국 을 먹었 습니다
```

나는 오늘 아침에 된장국을 먹었습니다.

I ate miso soup in this morning.

[('나', 'Noun'), ('는', 'Josa'), ('오늘', 'Noun'), ('아침', 'Noun'), ('에', 'Josa'), ('된장국', 'Noun'), ('을', 'Josa'), ('먹다', 'Verb'), ('.', 'Punctuation')]

나 오늘 아침 된장국 먹다

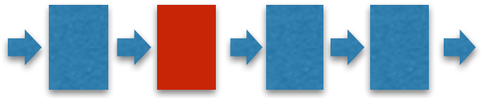
I / this morning / miso soup / eat

Data recycling / reusing

- Data recycling
  - Input of disintegrator → Output of grammar model
  - Output of disintegrator → Input of grammar model

```
original sentence     (output for grammar model): 그럼 다시 한 번 프로듀서 께서 소신 표명 을 해주시 겠 어요 ?
Disintegrated sentence (input for grammar model): 그렇다 다시 하다 번 프로듀서 소신 표명 해주다
original sentence     (output for grammar model): 저기 . 그러니까 .
Disintegrated sentence (input for grammar model): 저기 그러니까
original sentence     (output for grammar model): 프로듀서 로서 아직 경험 은 부족하지 만 아무튼 열심히 하겠 습니다 .
Disintegrated sentence (input for grammar model): 프로듀서 로서 아직 경험 부족하다 아무튼 열심히 하다
original sentence     (output for grammar model): 꿈 은 다 함께 톱 아이돌 !
Disintegrated sentence (input for grammar model): 꿈 다 함께 톱 아이돌
```

Conversation Bot model



- **Embedding RNN Sequence-to-sequence** model fo chit-chat
  - For testing purpose: 4-layer to 8-layer **swallow-learning** (without input/output layer)

What is **deep-learning model?**

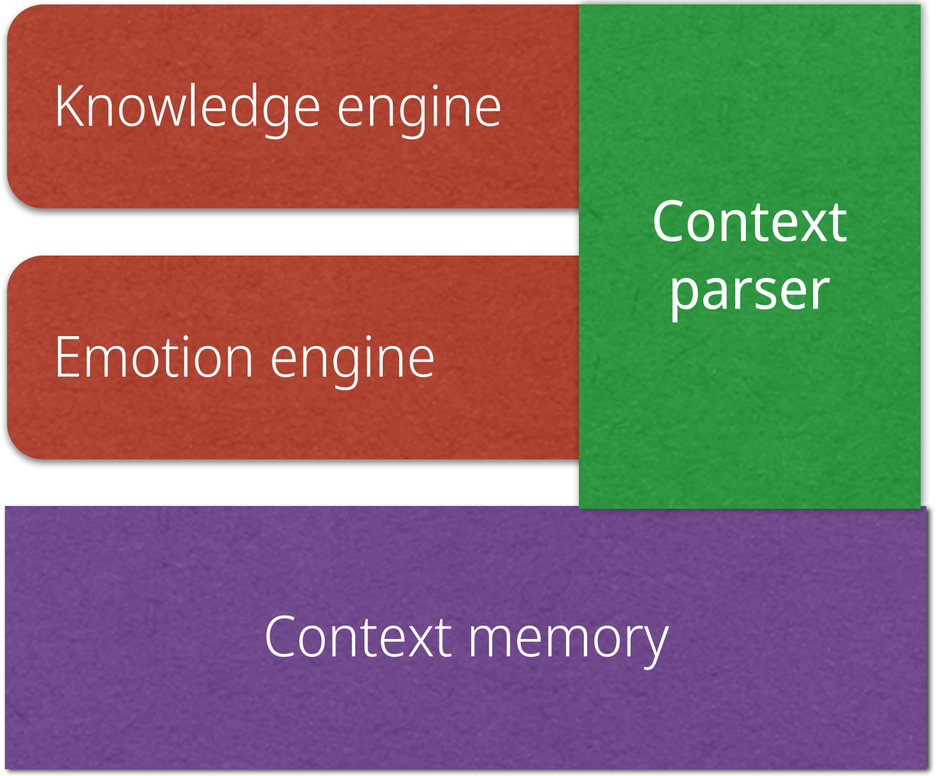
According review papers, ML with > 10 layers are.

And it's changing now... it became buzz word..

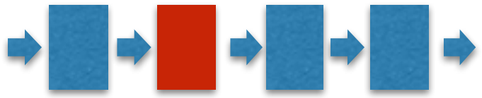
- Use **`tensorflow.contrib.learn`** (formally **`sklearn`** package)
  - Simpler and easier than traditional (3 month ago?) handcrafted RNN
  - Of course, **`seq2seq`**, **`LSTMCell`**, **`GRUCell`** are all bundled!

Context parser

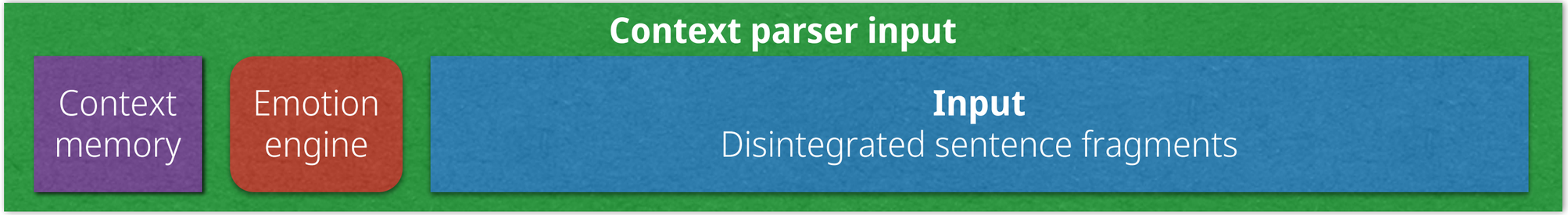
- Challenges
  - Continuous conversation
  - Context-aware talks
- Ideas
  - Context memory
  - Knowledge engine
  - Emotion engine



Memory and emotion

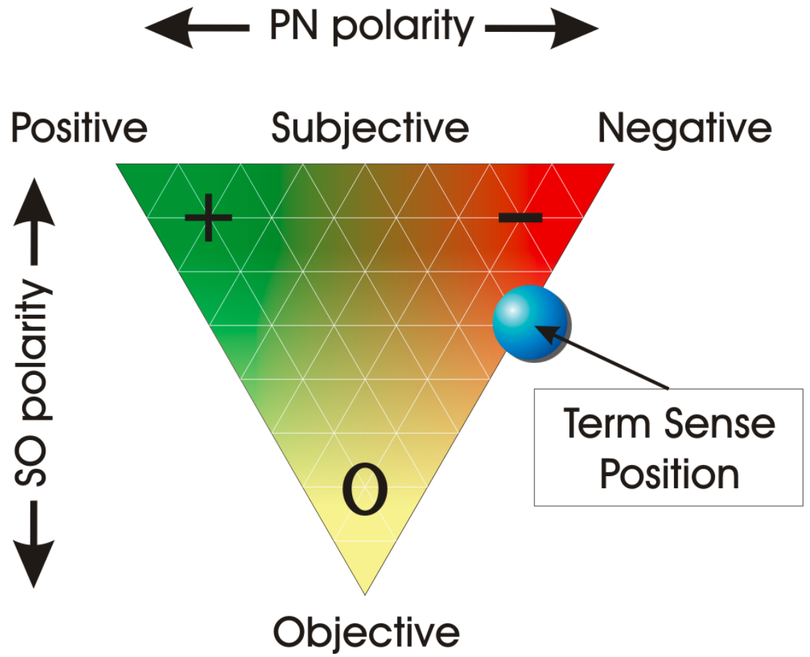


- Context memory as short-term memory
  - Memorizes current context (variable categories. Tested 4-type situations.)
- Emotion engine as model
  - Understands past / current emotion of user
- Use context memory / emotion engine as
    - First inputs of context parser model (for training / serving)



Emotion engine

- Input: *text sequence*
- Output: *Emotion flag (6-type / 3bit)*
- Training set
  - Sentences with 6-type categorized emotion
  - Uses **senti-word-net** to extract emotion
  - 6-axis emotional space by using **WordVec** model
  - **Current emotion indicator**: the most weighted emotion axis using **WordVec** model

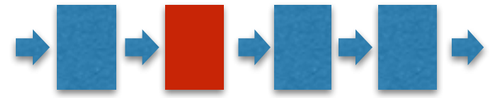


Position in senti-space:
[0.85, 0.14, 0.01, 0.05, 0.92, 0.23] → [1, 0, 0, 0, 0, 0] → 0x01

index: 1 2 3 4 5 6

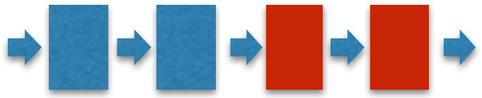
Illustration *(c) http://ontotext.fbk.eu/

Knowledge engine

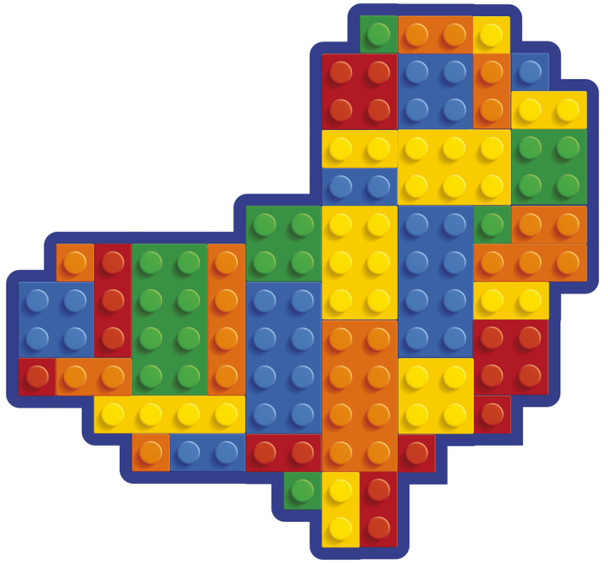


- Advanced topic: Not necessary for chit-chat bots
- Searches the tokenized knowledge related to current conversation
- Querying information
  - If target of conversation is query, use knowledge engine result as inputs of sentence generator
  - If information fitness is so high, ***knowledge+template*** shows great result
    - That's why information server bot will come to us soon at first.
- Big topic: **I'll not cover today.**

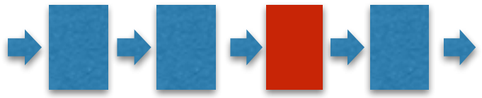
Sentence generator



- Generates human-understandable sentence as a reply of conversation
- Idea
  - Thinking and speaking is a “separate” processes in Brain
  - Why we use same model for these processes?
- Models
  - Consists of two models: Grammar generator + tone generator
- Why separate models?
  - Training cost
  - Much useful: various tones for user preferences

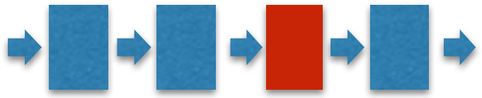


Grammar generator



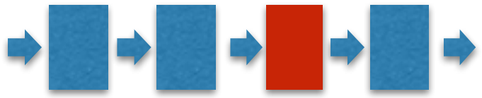
- Assembling sentence from word sequence
- Input: Sequence of Nouns, pronouns, verbs, adjectives
  - sentence without postpositions / conjunction.
- Output: Sequence of normal / monotonic sentence

Grammar generator



- Training set
  - Make sequence by disintegrating normal sentence
  - Remove postpositions / conjunction from sequence
  - Normalize nouns, verbs, adjectives
- Model
  - 3-layer Sequence-to-sequence model
  - Estimator: **ADAM** optimizer with **GRU** cell
    - **Adagrad** with **LSTM** cell is also ok. In my case, **ADAM**+**GRU** works slightly better. (Data size effect?)
  - *Hidden feature size* of GRU cell: 25, *Embedding dimension for each word*: 25.

RNN Seq2seq grammar model



Simple grammar model (word-based model with GRUCell and RNN Seq2seq / tensorflow translation example)

```
HIDDEN_SIZE = 25
EMBEDDING_SIZE = 25

def grammar_model(X, y):
    word_vectors = learn.ops.categorical_variable(X,
n_classes=n_disintegrated_words,
        embedding_size=EMBEDDING_SIZE, name='words')
    in_X, in_y, out_y = learn.ops.seq2seq_inputs(
        word_vectors, y, MAX_DOCUMENT_LENGTH, MAX_DOCUMENT_LENGTH)
    encoder_cell = tf.nn.rnn_cell.GRUCell(HIDDEN_SIZE)
    decoder_cell = tf.nn.rnn_cell.OutputProjectionWrapper(
        tf.nn.rnn_cell.GRUCell(HIDDEN_SIZE), n_recovered_words)
    decoding, _, sampling_decoding, _ = learn.ops.rnn_seq2seq(in_X, in_y,
        encoder_cell, decoder_cell=decoder_cell)
    return learn.ops.sequence_classifier(decoding, out_y, sampling_decoding)
```

Tone generator

- “Tones” to make sentence to be more humanized
- Every sentence has tones by speaker
- The most important part to build the “pretty girl chat-bot”
- Model
  - 3-Layer sequence-to-sequence model
  - Almost same as grammar model (training set is different)
- Can also be used to make chat bot speaking “dialects”

Tone generator

- **Input:** sentence without tones
- **Output:** sentence with tones
- **Data:** Normal sentences from various conversation sources
- Training / test set
  - Remove tones from normal sentences
    - morpheme treating effectively removes tone from sentence.

Useful tips

- **Sequence-to-sequence** model is *inappropriate* for Bot engine
  - Easily diverges during training
    - Of course, RNN training will not work.
    - in this case, input / output sequence relationship is too complex
  - Very hard to inject context-awareness to conversation
    - Response with context-aware need to ”generate” sentence not only from the ask, but with context-aware data / knowledgebase / decision making process
  - **Idea**: input sequence into **semantic bundle**
    - It will work, I guess…

Useful tips

- Sequence-to-sequence model really work well **with grammar / tone engine**
  - *This is **important** for today's.*

Training models

Goal is near here

Training bot model

- Input
  - Disintegrated sentence sequence without postpositions / conjunction
  - Emotion flag (3 bits)
  - Context flag (extensible, appending sentence with special indicator / 2 bits)
- Output
  - Answer sequence with nouns, pronouns, verbs, adjectives
- Learning
  - Supervised learning (for simple communication model / replaces template)
  - Reinforcement learning (for emotion / context flag, on the fly production)

Training bot model

- Training set
  - FAS log data (http://antispam.textcube.org)
    - 2006~2016 (from EAS data) / comments on weblogs / log size ~1TB (with spams)
    - Visited and crawled non-spam data, based on comment link (~26GB / MariaDB)
  - **Original** / **reply** pair as **input** / **output**
- Preprocessing
  - **Remove non-Korean characters** from data
  - **Data anonymization** with id / name / E-mail information

Training grammar generator

- Original data set
  - Open books without license problem (https://ko.wikisource.org)
  - Comments are not a good dataset to learn grammar
- Preprocessing
  - **Input data**: disintegrated sentence sequence
  - **Output data**: original sentence sequence

Training tone generator

- Original data set
  - Open books without license problem
  - Extract sentences wrapped with “
    - e.g. "집에서 온 편지유? 무슨 걱정이 생겼수?"
- Preprocessing
  - **Input data**: sentence sequence without tone
    - e.g. “집에서 온 편지? 무슨 걱정 생기다?” (using morpheme analyzer)
  - **Output data**: original sentence sequence

One page summary

The simplest is the best

설마 날 신경써주고 있는 거야?

Disintegrator

설마 날 신경 써주다 있다

Context analyzer

[GUESS] 날 [CARE] [PRESENT]

Decision maker

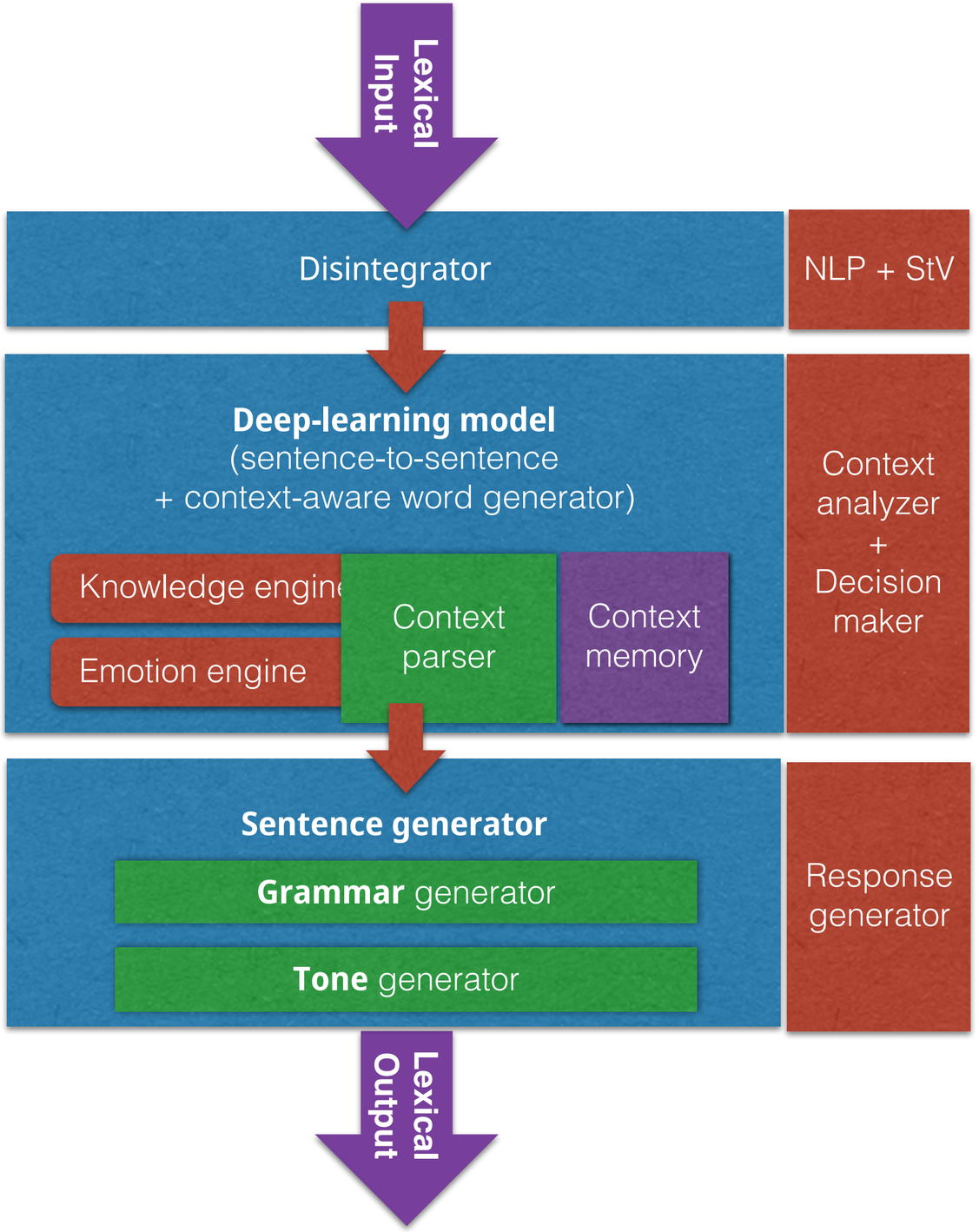
어제 네 기운 없다

Grammar generator

어제 네가 기운이 없길래

Tone generator

어제 네가 기운이 없길래 요



No way, are you caring me now?

Disintegrator

no way you care I now

Context analyzer

[GUESS] I [CARE] [PRESENT]

Decision maker

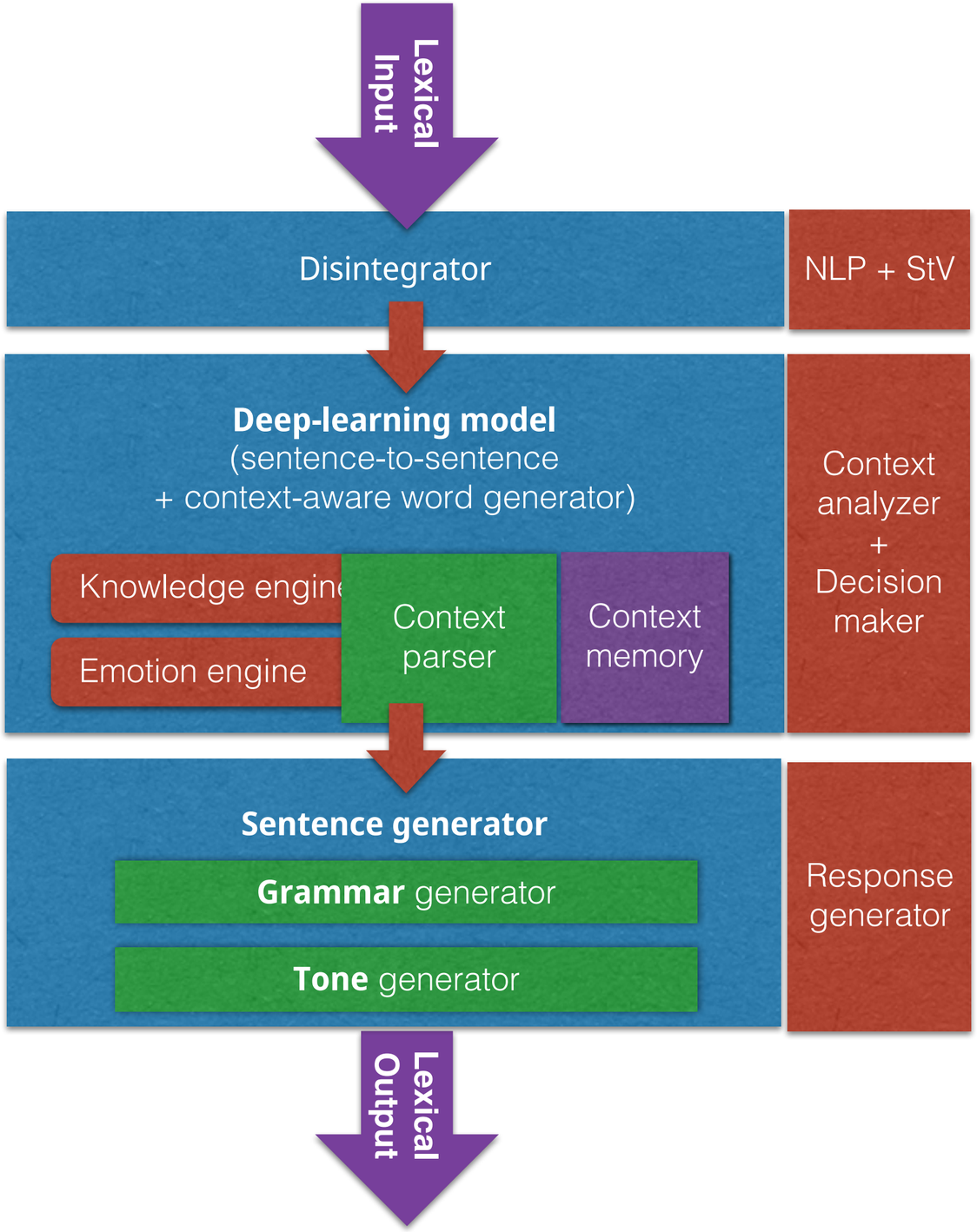
because yesterday you tired

Grammar generator

Because you looked tired yesterday

Tone generator

Because you looked tired yesterday hmm



But this is **not** what I **promised**...

at PyCON APAC abstract

Making 미소녀bot

Let's make ***anime character bot*** (as I promised)!

Data source

- ***Subtitle (caption) files*** of many ***Animations***!
- Prototyping
  - Idol master conversation script (translated by online fans)
- Field tests
  - Animations only with female characters

Data converter

.smi to .srt

Join
.srt files into one .txt

```
cat *.srt >> data.txt
```

Remove
timestamps
and
blank lines

Remove
Logo / Ending
Song scripts

: Lines with
Japanese
Characters
and
the next lines
of them

Fetch

Character names
Nouns
Numbers
using
custom dictionary

(Anime characters,
Locations,
Specific nouns)

subtitle_converter.py

*.smi file format is *de facto standard* of movie caption files in Korea

Reformat

merge
sliced captions
into one line

Remove

Too short sentences
Duplicates

Sentence data
for

disintegrator
grammar model
tone model

pandas

Train

disintegrator
integrator with
grammar model
tone model

Extract Conversations

```
if last_sentence [-1] == '?':
  conversation.add((
    last_sentence,
    current_sentence))
```

subtitle_converter.py

Conversation data
for sequence-to-sequence
Bot model

pandas

Train

bot model

Conveniences for demo

- Simple bot engine
  - *ask* – *response* sentence similarity match engine (similar to template engine)
- Merge grammar model with tone model
  - Grammar is not important to create anime character bot?
- Loose parameter set
  - For fast convergence: data size is not big / too diverse
- No knowledge engine
  - We just want to talk with him/her.

Bot training procedure (initialization)

```
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcudnn.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcuda.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcurand.so locally
total conversations: 4217
Transforming...
Total words, asked: 1062, response: 1128
Steps: 0
I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:924] successful NUMA node read from SysFS had
negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
I tensorflow/core/common_runtime/gpu/gpu_init.cc:102] Found device 0 with properties:
name: GeForce GTX 970
major: 5 minor: 2 memoryClockRate (GHz) 1.304
pciBusID 0000:01:00.0
Total memory: 4.00GiB
Free memory: 3.92GiB
I tensorflow/core/common_runtime/gpu/gpu_init.cc:126] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:136] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:806] Creating TensorFlow device (/gpu:0) -> (device:
0, name: GeForce GTX 970, pci bus id: 0000:01:00.0)
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:244] PoolAllocator: After 1501 get requests,
put_count=1372 evicted_count=1000 eviction_rate=0.728863 and unsatisfied allocation rate=0.818787
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:256] Raising pool_size_limit_ from 100 to 110
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:244] PoolAllocator: After 2405 get requests,
put_count=2388 evicted_count=1000 eviction_rate=0.41876 and unsatisfied allocation rate=0.432432
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:256] Raising pool_size_limit_ from 256 to 281
```

Bot model training procedure (after first fitting)

```
ask: 시 분 시작 하다 이 것 대체 <REP>.
response (pred): NAME 해오다 <REP>.
response (gold): NAME 죄송하다.

ask: 쟤 네 <UNK> 사무소 주제 너무 <UNK> 하다 거 알다.
response (pred): NAME 해오다 <REP>.
response (gold): 아깝다 꼴 찌다 주목 다 받다

ask: <UNK> 아니다 <REP>.
response (pred): NAME 해오다 <REP>.
response (gold): 더 못 참다
```

Bot model training procedure (after 50 more fittings)

```
ask: 이렇다 상태 괜찮다 <REP>.
response (pred): 이렇다 여러분 <REP>.
response (gold): NOUN 여러분.

ask: 기다리다 줄 수 없다 <REP>.
response (pred): 네 충분하다 기다리다 <REP>.
response (gold): 네 충분하다 기다리다.

ask: 넌 뭔가 생각 하다 거 있다 <REP>.
response (pred): 물론 이 <REP>.
response (gold): 물론 이.
```

Trust me.
Your **NVIDIA card** can not only play Overwatch, but this, too.

Grammar+Tone model training procedure (initialization)

```
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcudnn.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcuda.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened CUDA library libcurand.so locally
total line: 7496
Fitting dictionary for disintegrated sentence...
Fitting dictionary for recovered sentence...
Transforming...
Total words pool size: disintegrated: 3800, recovered: 5476
I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:924] successful NUMA node read from SysFS had
negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
I tensorflow/core/common_runtime/gpu/gpu_init.cc:102] Found device 0 with properties:
name: GeForce GTX 970
major: 5 minor: 2 memory
ClockRate (GHz) 1.304
pciBusID 0000:01:00.0
Total memory: 4.00GiB
Free memory: 3.92GiB
I tensorflow/core/common_runtime/gpu/gpu_init.cc:126] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:136] 0:   YI
tensorflow/core/common_runtime/gpu/gpu_device.cc:806] Creating TensorFlow device (/gpu:0) -> (device: 0,
name: GeForce GTX 970, pci bus id: 0000:01:00.0)
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:244] PoolAllocator: After 1501 get requests,
put_count=1372 evicted_count=1000 eviction_rate=0.728863 and unsatisfied allocation rate=0.818787
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:256] Raising pool_size_limit_ from 100 to 110
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:244] PoolAllocator: After 2405 get requests,
put_count=2388 evicted_count=1000 eviction_rate=0.41876 and unsatisfied allocation rate=0.432432
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:256] Raising pool_size_limit_ from 256 to 281
```

Grammar+Tone model training procedure (after first fitting)

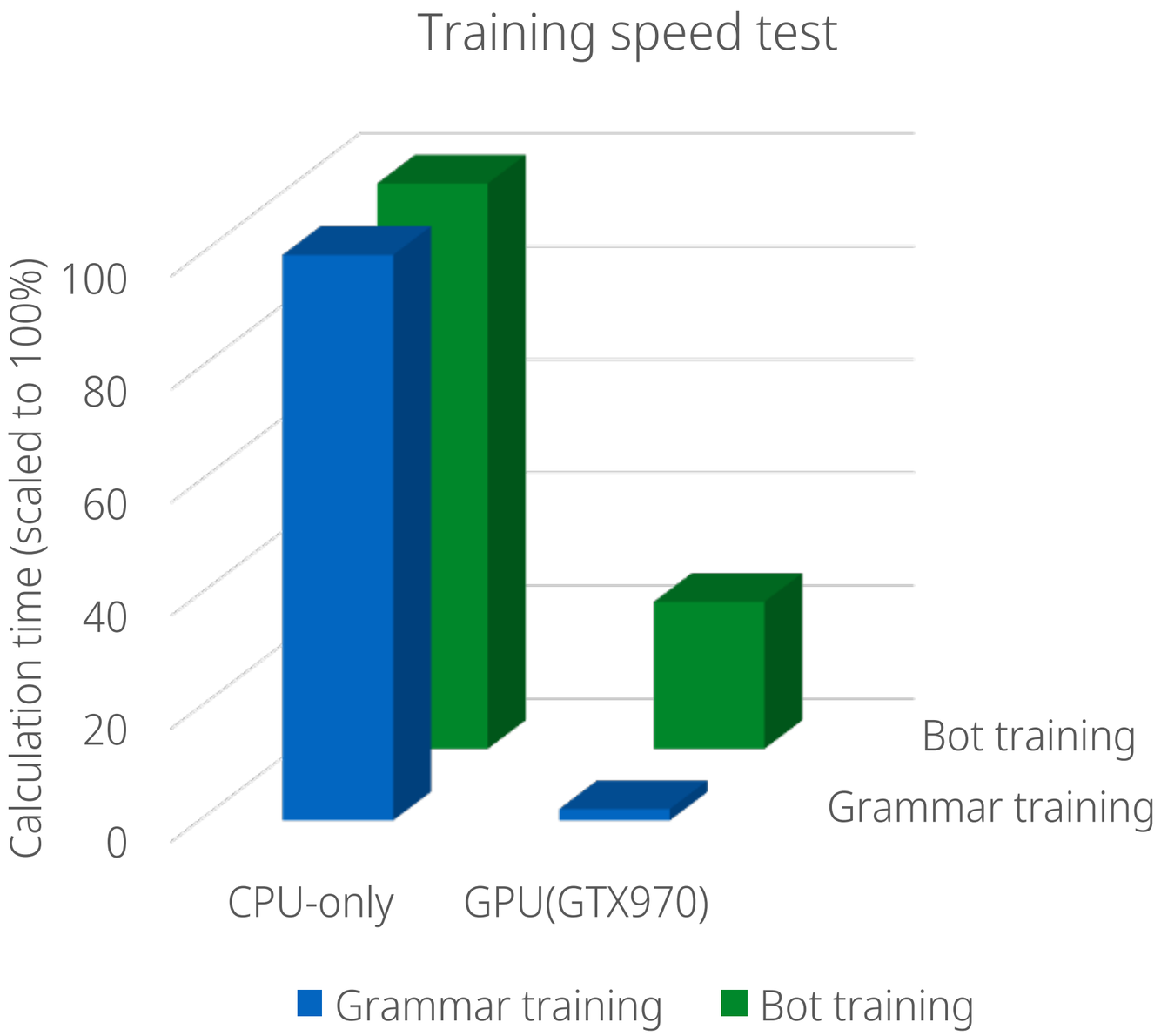
```
disintegrated: 올해 우리 프로덕션 NOUN 의 활약 섭외 들어오다 <REP>.
recovered (pred): 그래서 저기 들 나요 <REP>.
recovered (gold): 올해 는 우리 프로덕션 도 NOUN 의 활약 으로 섭외 가 들어왔 답 니다.
disintegrated: 둘 다 왜 그렇다 <REP>.
recovered (pred): 어머 어머 아 <REP>.
recovered (gold): 둘 다 왜 그래.
disintegrated: 정말 우승 하다 것 같다 <UNK> .
recovered (pred): 정말 를 <REP>.
recovered (gold): 정말 우승할 것 같네 요.
disintegrated: 아 진짜 <REP>.
recovered (pred): 아 아 을까 <REP>.
recovered (gold): 아 진짜.
```

Grammar+Tone model training procedure (after 10 more fitting)

```
disintegrated: 호흡 딱 딱 맞다 <REP>.
recovered (pred): 무슨 을 <REP>.
recovered (gold): 호흡 이 딱 딱 맞 습니다.
disintegrated: 무슨 소리 NAME <REP>.
recovered (pred): 무슨 소리 음 <REP>.
recovered (gold): 무슨 소리 야 NAME.
disintegrated: 너 맞추다 또 넘어지다 거 잖다 <UNK> <UNK> <UNK> <UNK>.
recovered (pred): 너 겹친 또 넘어질 거 <REP>.
recovered (gold): 너 한테 맞춰 주 면 또 넘어질 거 잖아.
disintegrated: 중계 나름 신경 써주다 <REP>.
recovered (pred): 무대 에서도 을 신경 <REP>.
recovered (gold): 중계 에서도 나름 대로 신경 을 써줘.
```

Grammar model converges fast.

With GPU, it converges much faster.



And you must need
GPU-accelerated environment
to let them work.

Useful tips for anime character bot

- DO NOT MIX different anime subtitles
  - Easily diverges during grammar model training. Strange. Huh?
  - Does it come from different translator's tone? Need to check why.
- Choose animation with extreme gender ratio
  - Very hard to divide gender-specific conversations from data
  - Tones of Japanese animation character are very different by speakers' gender
  - Just choose boy-only / girl-only animation for easy data categorization

And tackles today

- From TensorFlow 0.9RC, `Estimator/TensorFlowEstimator.restore` is removed and not returned yet
  - I can create / train model but cannot load model with original code on TF 0.10RC.
- Made some tricks for today's demo
  - Auto-generated talk templates from bot
  - Response matcher (match ask sentence and return response from template pool)
- Conversation dataset size is too small to create conversation model
  - Not smooth talks
  - Easily diverges. Train many, many models to get proper result.

Serving

Like peasant in Warcraft (OR workleft?)

Telegram API

- Why Telegram?
  - Telegram is my primary messenger
  - API implementation is as easy as writing echobot
  - Well-suited with python 3

Serving Telegram bot

- Python 3

Install python-telegram-bot package

```
~$ pip3 install python-telegram-bot
```

- Supervisor (for continuous serving)

/etc/supervisor/conf.d/pycon_bot.conf

```
[program:pycon-bot]
command = /usr/bin/python3 /home/ubuntu/pycon_bot/serve.py
```

supervisorctl

```
ubuntu@ip-###-###-###-###:~$ sudo supervisorctl
pycon-bot                        RUNNING    pid 12417, uptime 3:29:52
```

Bot serving code

/home/ubuntu/pycon_bot/serve.py

```
from telegram import Updater
from pycon_bot import pycon_bot, error, model_server

bot_server = None
grammar_server = None

def main():
    global bot_server, grammar_server
    updater = Updater(token='[TOKENS generated via bot_father]')
    job_queue = updater.job_queue
    dispatcher = updater.dispatcher
    dispatcher.addTelegramCommandHandler('start', start)
    dispatcher.addTelegramCommandHandler("help", start)
    dispatcher.addTelegramMessageHandler(pycon_bot)
    dispatcher.addErrorHandler(error)
    bot_server = model_server('./bot', 'ask.vocab', 'response.vocab')
    grammar_server = model_server('./grammar', 'fragment.vocab', 'result.vocab')
    updater.start_polling()
    updater.idle()

if __name__ == '__main__':
    main()
```

Model server

pycon_bot.model_server

```
class model_server(self):
    """ pickle version of TensorFlow model server """
    def __init__(self, model_path='.', x_proc_path='', y_proc_path=''):
        self.classifier = learn.TensorFlowEstimator.restore(model_path)
        self.X_processor = pickle.loads(open(model_path+'/'+x_proc_path,'rb').read())
        self.y_processor = pickle.loads(open(model_path+'/'+y_proc_path,'rb').read())

    def predict(input_data):
        X_test = X_processor.transform(input_data)
        prediction = self.classifier.predict(X_test, axis=2)
        return self.y_processor.reverse(prediction)
```

Bot engine code

pycon_bot.pycon_bot

```
def pycon_bot(bot, update):
    msg = disintegrate(update.message.text)
    raw_response = bot_server.predict(msg)
    response = grammar_server.predict(raw_answer)
    bot.sendMessage(chat_id=update.message.chat_id, text=' '.join(response))
```

pycon_bot.disintegrate

```
def disintegrate(sentence):
    disintegrated_sentence = konlpy.tag.Twitter().pos(sentence, norm=True,
stem=True)
    result = []
    for w, t in disintegrated_sentence:
        if t not in ['Eomi', 'Josa', 'Number', 'KoreanParticle', 'Punctuation']:
            result.append(w)
    return ' '.join(result)
```

Result

That's one small step for a man, one giant leap for anime fans.

Hi

When will we open this bot to public?

Sorry Jeongkyu.

Sorry? Why?

I apologize to seniors

;;;

[ERROR]

The weather is so hot.

Suddenly but I feel sorry

What makes you feel like that?

Nowadays I lose my concentration.

Ah. sometimes I do too.

Getting stressful?

My work is very stressful.

Let’s not be nervous today.

I’m still nervous.

And finally... created pretty sad bot.

Reason?

Idol M@ster's conversations are mostly about failure and recover rather than success.

Summary

- Today
  - Covers garage chat bot making procedure
  - Making chat bot with **TensorFlow** + Python 3

- My contributions / insight to you
  - **Multi-modal** Learning models / structures for chat-bots
  - **Idea to generate “data”** for chat-bots

And next...

- Suggestion from ***Shin Yeaji*** (PyCon APAC staff) and ***my wife*** in this week
  - Train bot with some unknown (to me) animations.
- Finish anonymization of FAS data and re-train bot with ***TensorFlow***
  - In fact, FAS data-based bot is run by ***Caffe***. (http://caffe.berkeleyvision.org/)
  - This speak preparation encourages me to migrate my ***Caffe*** projects to ***TensorFlow***
- Test Seq2seq to bot engine?
  - By making input sequence into **semantic bundle**

Home assignment

- If you are Loveliver*, you already know what to do.

*The fans of *lovelive* (another Japanese animation)

Internet meme * (c) Marble Entertainment / inven.co.kr

Home assignment

- If your native language is English, how about making

?

“First with the head, then with the heart.”

–Bryce Courtenay's *The Power of One*

Thank you for listening :)

@inureyes

Slides available via pycon.kr
Code will be available at https://github.com/inureyes/pycon-apac-2016

Selected references

- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013, January 17). Efficient Estimation of Word Representations in Vector Space. *arXiv.org*.
- Schmitz, C., Grahl, M., Hotho, A., & Stumme, G. (2007). Network properties of folksonomies. *World Wide Web ...*.
- Esuli, A., & Sebastiani, F. (2006). Sentiwordnet: A publicly available lexical resource for opinion mining. Presented at the Proceedings of LREC.
- De Brabandere, B., Jia, X., Tuytelaars, T., & Van Gool, L. (2016, June 1). Dynamic Filter Networks. *arXiv.org*.
- Noh, H., Seo, P. H., & Han, B. (2015, November 18). Image Question Answering using Convolutional Neural Network with Dynamic Parameter Prediction. *arXiv.org*.
- Andreas, J., Rohrbach, M., Darrell, T., & Klein, D. (2015, November 10). Neural Module Networks. *arXiv.org*.
- Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. (2015, June 10). Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *arXiv.org*.
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science (New York, NY)*, *349*(6245), 253–255. http://doi.org/10.1126/science.aac4520
- Bahdanau, D., Cho, K., & Bengio, Y. (2014, September 2). Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv.org*.
- Schmidhuber, J. (2014, May 1). Deep Learning in Neural Networks: An Overview. *arXiv.org*. http://doi.org/10.1016/j.neunet.2014.09.003
- Zaremba, W., Sutskever, I., & Vinyals, O. (2014, September 8). Recurrent Neural Network Regularization. *arXiv.org*.
- Smola, A., & Vishwanathan, S. V. N. (2010). Introduction to machine learning.