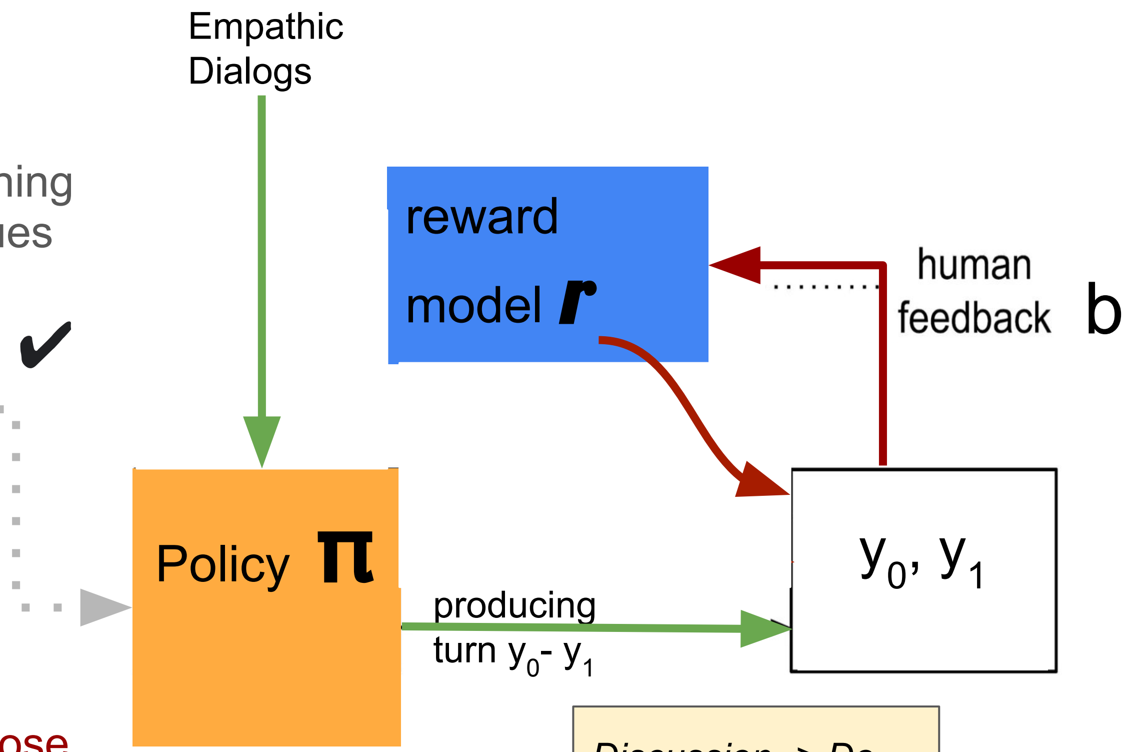
What did we do?

Our Vision - Part 1

1. Creating a Baseline: Fine-Tuning GPT-2 with Empathic Dialogues datasets
   a. Data Preprocessing (Encoding, Decoding & Importing Data) ✔
   b. Fine-Tuning to get π
2. Producing turns y_0- y_1 to give human feedback on
   a. *With test/validation set of*

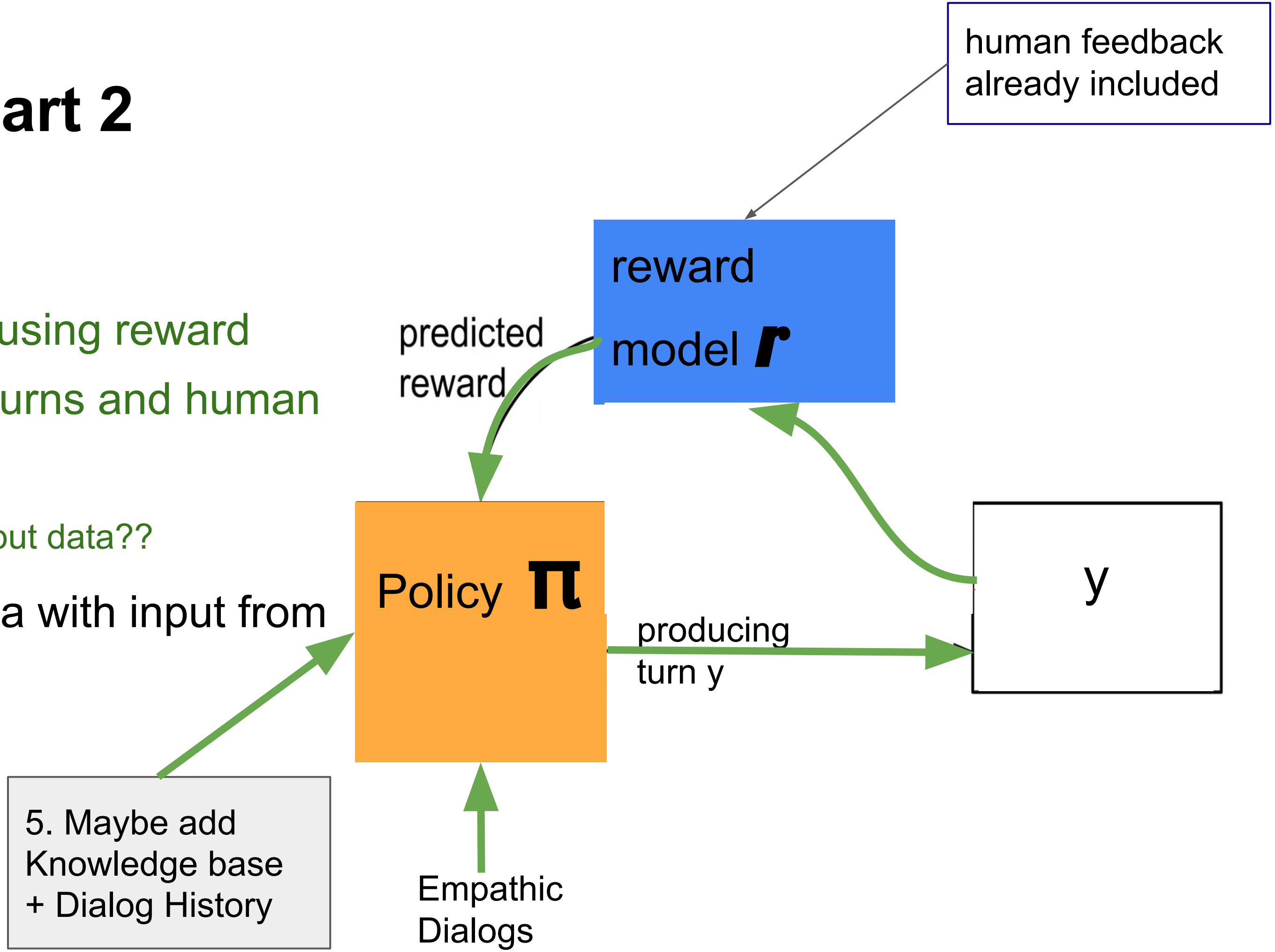
      Empathic Dialogs dataset
3. Train reward model r with those turns and human feedback b



Discussion -> Do we want to use 2/3/4 comparisons?

Our Vision - Part 2

4. Train policy π using reward model r with those turns and human feedback b
    a. With which input data??
5. Improve the idea with input from other papers



Preprocessing EmpathicDialogues (Step 1.a)

What we use:

	A	B	C	D	E	F	G	H
1	conv_id	utterance_id	context	prompt	speaker_idx	utterance	selfeval	tags
2	hit:0_conv:1	1	sentimental	I remember going to	1	I remember going to see the fireworks with my best friend. It was the	5|5|5_2|2|5	
3	hit:0_conv:1	2	sentimental	I remember going to	0	Was this a friend you were in love with_comma_ or just a best friend?	5|5|5_2|2|5	
4	hit:0_conv:1	3	sentimental	I remember going to	1	This was a best friend. I miss her.	5|5|5_2|2|5	
5	hit:0_conv:1	4	sentimental	I remember going to	0	Where has she gone?	5|5|5_2|2|5	
		5	sentimental	I remember going to	1	We no longer talk.	5|5|5_2|2|5	
		6	sentimental	I remember going to	0	Oh was this something that happened because of an argument?	5|5|5_2|2|5	
		1	afraid	i used to scare for d	2	it feels like hitting to blank wall when i see the darkness	4|3|4_3|5|5	

Change: Better loss if we add the <SOC> token only once prior to the last turn

🙂 we have 1 - 5 previous turns as context

~ 64 000 Trainingsamples

I remember going to see the ….

<SOC> Was this a friend you were in love …

<EOT> #########

Trainingsample 1

I remember going to see the ….

<SOC> Was this a friend you were in love …

<SOC> This was a best friend. I miss her.

<EOT> #########

Trainingsample 2

I remember going to see the ….

<SOC> Was this a friend you were in love …

<SOC> This was a best friend. I miss her.

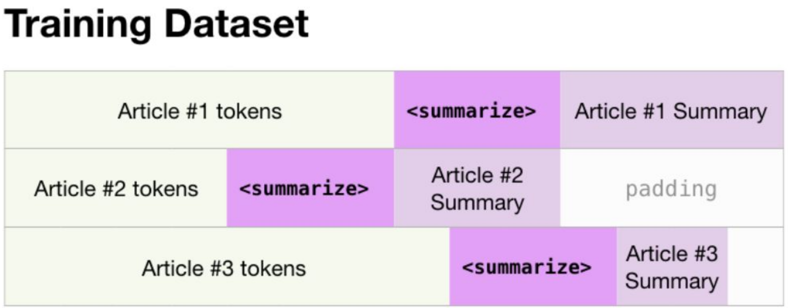
<SOC> Where has she gone?

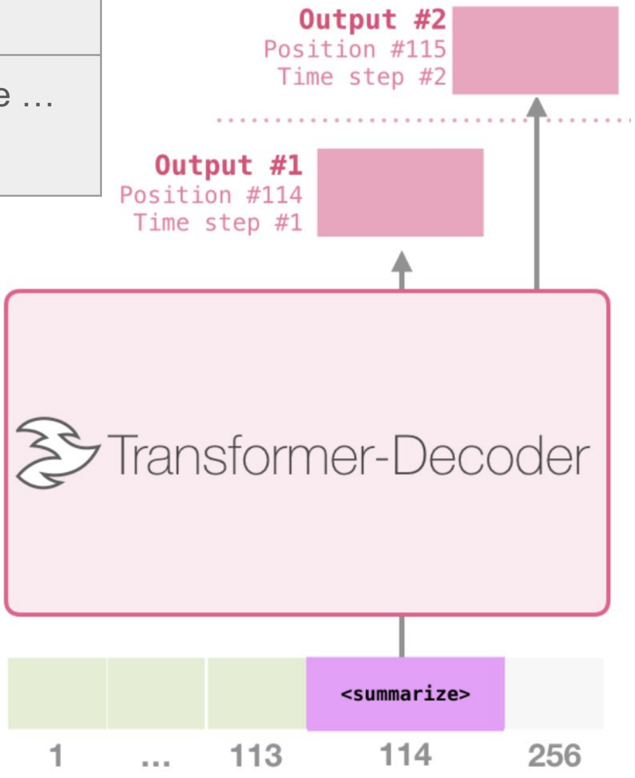
<EOT> ##########################

Trainingsample 3

Transfer learning like described in “**The Illustrated GPT-2** ”

context	separator	response/turn
I remember going to see the ….	<SOC>	Was this a friend you were in love …





Thinking about labelling/rating ...

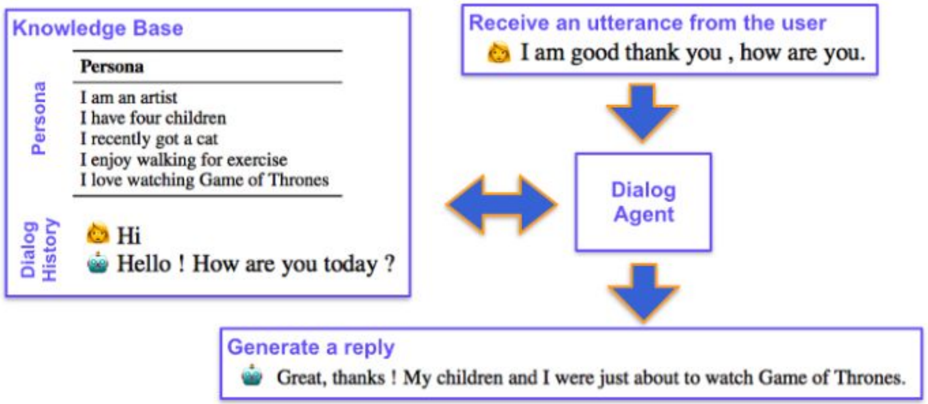
When?

- Step 3 - Label which turn/response is better; 2/3/4 comparisons?
- General model rating - Do we compare our models by human rating?

How?

- What is a “better” turn/response?
- Empathy/Sympathy: did the responses show understanding of the feelings of the person talking about their experience?
- Relevance: did the responses seem appropriate to the conversation? Were they on-topic?
- Fluency: could you understand the responses? Did the language seem accurate?

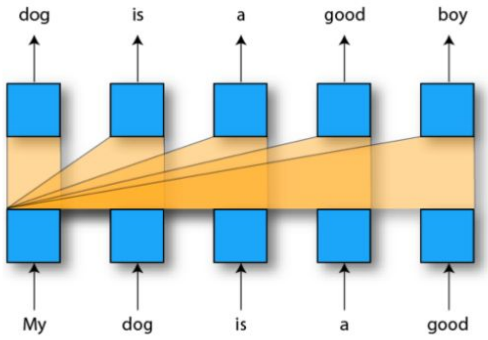
Conversational AI with a persona



Knowledge base stores information about persona and dialog history

Receiving new utterance
→ agent combines content of this knowledge base with the newly received utterance to generate *a* reply

using transfer learning:
→ gain storing knowledge while solving one problem and applying it to a different but related problem



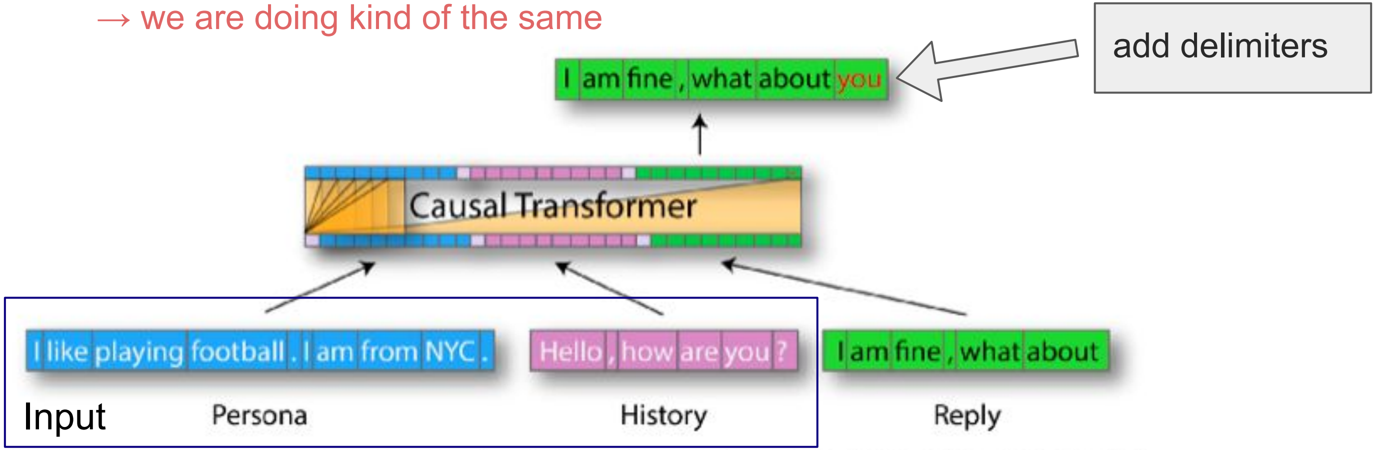
Adapting a language model to a dialog task

How can we build an input for our model from these various contexts?

- ➢ **concatenate(verketten)** the context segments in a single sequence, putting the reply at the end
    - **issues**: transformer is color & position blind
    - **solution:** add position information for each token

      → implementation by adding special tokens to our vocabulary for delimiters and segment indicators with pytorch-pretrained-BERT classes

      → we are doing kind of the same



Input sequence: a concatenation of persona (blue), history (pink) and reply (green) with delimiters (light pink).
Here we generate the word "you" to complete the reply.

State-based Dialog Modeling Using Human Preferences,

Master Thesis Florian von Unold

goal: control data driven dialog generation w.r.t. human communication characteristics

State of the art language models

Task

Learn statistical dependencies of words on giant amount of unsupervised text data → predict the next word (token) given the surrounding context

Problem

- Difficult to **control the model output**
- Lack of method to **include human preferences**

Idea

- Collect human feedback on the quality of generated dialogs
- Use this feedback directly to optimize the model

Issues

- Lots of feedback required
  → Very time consuming process

Florian’s Projects

1. **Learning dialog from human preferences:**

   Preliminary study to using human feedback data to learn a reward function that can be optimized with Reinforcement Learning

- ➔ humans decide for two dialogs which dialog they prefer
- ➔ problem: time consuming
- ➔ solution approach: Possibly doable with many crowdworkers in parallel (means? --> we have no money to hire people → but friends & family)
- ➔ use his skript for annotating → maybe ask him for an introduction meet up to his code

2. **State based dialog modeling:**

   Learn the dynamics of utterance metrics in human dialogs with a neural network (DYME - a DYnamic MEtric) which can then be used to optimize pre-trained dialog models with Reinforcement Learning

- ➔ BERT-based approach to model the change of utterance metrics within dialogs (DYME)
  - → train a neural network that predicts the metrics of the next sentence given a dialog history
  - → fine-tune a pre-trained VHRED model with VHRL using the deviation from DYME as a negative reward signal
  - → All metrics can be computed on two human-written, bi-turn dialog datasets → Further matrices: empathy mechanisms

Florian’s Research & Results

Hypothesis	Experiment	Conclusion
Metrics in human dialogs are dynamic	Quantify (utterance-level) dynamics of metrics within dialogs on two open-domain dialog datasets → result?	Metrics in human dialogs are dynamic
A dynamic metric can be learned from dialog data (**dy**namic **me**tric = DYME)	Learn a dynamic metric (DYME) from dialog data → DYME improves over baseline (mean)	A dynamic metric can be learned from dialog data (**dy**namic **me**tric = DYME)
Using DYME in (VH)RL fine-tuning of an open-domain chatbot improves the chatbot‘s „performance“	Compare VHRL fine-tuning with DYME to VHRED baseline (starting from same pre-trained model) → Preliminary results: DYME **and** baseline not convincing qualitatively → Searching for a good baseline	RL fine-tuning of an open-domain chatbot is a promising approach, however solid baseline missing for evaluation

Calculating utterance reward

utterance reward = 0.15 ∗ sentiment + 0.25 ∗ question + 0.5 ∗ repetition + 0.05 ∗ similarity + 0.05 ∗ toxicity

-question: positive reward (+1) if the current utterance contains a question word and a question mark
-sentiment: higher reward for higher sentiment (lower reward for lower sentiment)

→ According to the reward, every utterance at any given position in any dialog should adhere to these metrics and coefficients

Example dialog:
[Speaker 1]: Hey!
[Speaker 2]: Hi, how are you?
[Speaker 1]: Fine, and you?
[Speaker 2]: I just finished watching a very bad movie… → suitable utterance in proper language will receive lower reward → WHY?
I thought suitable utterance is good → reward higher?!

→ **Possible problem:** static rewards, overall utterance reward given by the developer-defined equation (means?!)

Our codes (from original GPT2) (just for my understanding)

01_process_ data: pretraining our input text data

1. load_dataset:
- dataset located in empathicdialogues/train.scv (many dialooogues with sith sentiment and context included)
- sentence to tensor (context, response, encoder, maxlen)= encoding part → give input to model that model understand the tect → transform sentence into arra/matrix with numbers : “Hi”=13750, “how”=11

```
a = "Hi, How are you?"
b = "Hi how are you" + " <SOC> "
model = "I am fine. <EOT>"
```

```
print(enc.encode(a))
```

```
[17250, 11, 1374, 389, 345, 30]
```

```
print(enc.decode([17250, 11, 1374, 389, 345, 30]))
Hi, How are you?
```

- decoder= transforms numbers iinto words
- replace “comma” with “,” ,
- SOC=”Start of Conversation” stands at the end of the input context/previous dialog → +”SOC” to tell the model that we need a reply
- <EOT>=”End of Text”
- Use decoder to learn the model from the input dialogs what is the sentence and the follwoing response and so on → context is increasing the further the conversation is
- Use encoder afterwards to
- putting training samples togehter which belong together → always 5 sentence are one conversation

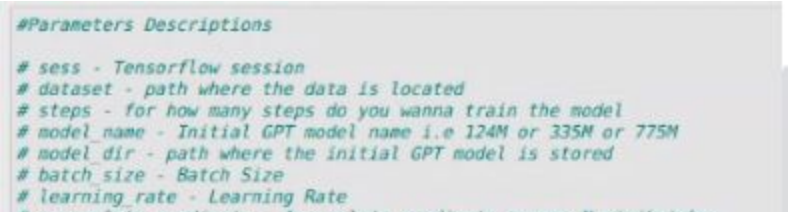
- transform every dialog sentence in one matrix? including the previous context?
- dont save preprocced data → just call it “def_process_training_data
- fine tuning: optimizes the model → repsonse suitable aswers / / depending on the sentiment and empathic of the human
- - transfer learning= train model on images (cat&dog)

```
print(enc.decode(data[0]))
print("\n")
print(enc.decode(data[1]))
print("\n")
print(enc.decode(data[2]))
print("\n")
print(enc.decode(data[3]))
print("\n")
print(enc.decode(data[4]))
print("\n")
print(enc.decode(data[5]))
```

```
I remember going to see the fireworks with my best friend. It was the first time we ever spent time alone togethe
r. Although there was a lot of people, we felt like the only people in the world. <SOC> Was this a friend you were
in love with, or just a best friend? <EOT> ################################

I remember going to see the fireworks with my best friend. It was the first time we ever spent time alone togethe
r. Although there was a lot of people, we felt like the only people in the world. <SOC> Was this a friend you were
in love with, or just a best friend? <SOC> This was a best friend. I miss her. <EOT> ##################

I remember going to see the fireworks with my best friend. It was the first time we ever spent time alone togethe
r. Although there was a lot of people, we felt like the only people in the world. <SOC> Was this a friend you were
in love with, or just a best friend? <SOC> This was a best friend. I miss her. <SOC> Where has she gone? <EOT> ###
######

I remember going to see the fireworks with my best friend. It was the first time we ever spent time alone togethe
r. Although there was a lot of people, we felt like the only people in the world. <SOC> Was this a friend you were
in love with, or just a best friend? <SOC> This was a best friend. I miss her. <SOC> Where has she gone? <SOC> We
no longer talk. <EOT>

Was this a friend you were in love with, or just a best friend? <SOC> This was a best friend. I miss her. <SOC> Wh
ere has she gone? <SOC> We no longer talk. <SOC> Oh was this something that happened because of an argument? <EOT>
################################

it feels like hitting to blank wall when i see the darkness <SOC> Oh ya? I don't really see how <EOT> ############
#########################################################
```

Finetuning : pre_train_gpt2.py

- import the context = everything standing before <SOC>
- tensor flow
- loss function of language model= tries to teach the model what is correct and what is not, after “I” not another subject will follow → loss high
  after “I” the verb “am” will follow is more likely to be → loss low
  (given the context & loss function→ predict next token)
- optimize function: gradient descent with momentum → we try to minimize the loss function
- def save : to save the model after 1000 steps of improving/after 1000 losses
- def finetune:



training part in pre_train_gpt2

```
def sample_batch():
    return random.sample(data, batch_size)

if overwrite and restore_from == 'latest':
    for file in files:
        if file.startswith('model') or file.startswith('events'):
            os.remove(os.path.join(checkpoint_path, file))
    save()

avg_loss = (0.0, 0.0)
start_time = time.time()

if steps:
    steps = int(steps)

try:
    while True:
        if steps > 0 and counter == (counter_base + steps):
            save()
            return
        if (counter - 1) % save_every == 0 and counter > 1:
            save()

        sess.run(opt_reset)
        for _ in range(accumulate_gradients):
            sess.run(
                opt_compute, feed_dict={context: sample_batch()})
        (v_loss, v_summary) = sess.run((opt_apply, summary_loss))

        summary_log.add_summary(v_summary, counter)

        if counter % print_every == 0:
            avg_loss = (avg_loss[0] * 0.99 + v_loss,
                        avg_loss[1] * 0.99 + 1.0)

            print(
                '[{counter} | {time:2.2f}] loss={loss:2.2f} avg={avg:2.2f}'
                .format(
                    counter=counter,
                    time=time.time() - start_time,
```

sample_batch= import dialog input

- we have 64615 sentences → 5 together are one conversation → we have 64615/5 different dialogs from “EmapthicDialogs” saved in “data”

Next Step 1.b Finetuning to get pi

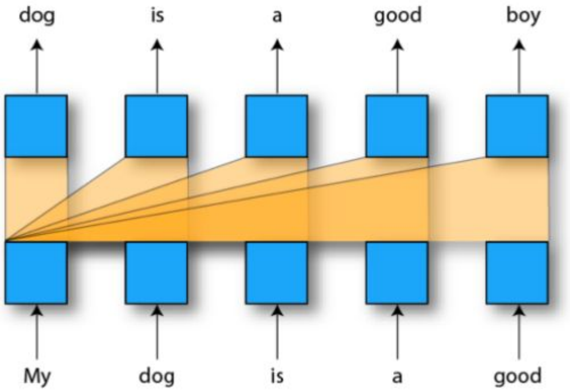
- we want to recognize the emotions, human sentiments and train model
- write sample sequences
- we are fine tuning the original gpt2 model
- use dialogs from validation set and sample two responses→ save it in file → annotate it later → use for policy

pre_train_gpt2 = for fine tuning

model saved in “checkpoint”

Transfer Learning <u>**How to build a State-of-the-Art Conversational AI with Transfer Learning, Wolf**</u>

used to build a State of the Art dialog agent based on **OpenAI GPT** and **GPT-2** Transformer language models



goal: gain storing knowledge while solving one problem and applying it to a different but related problem

Next Steps

- Writing the annotation script for training reward model with human feedback
- optimize Fine Tuning
- how we sample the context:
- for given context the function samples_sequences samples the responses
  → write similar function to feed dialog to the model
  → generate 2 responses for given context
- use zieglers: lm-human-preferences/lm_human_preferences/language/sample.py “sample_sequence function” and improvise that we generate 2 responses and not just only one
- write new script and call it in the notebook to test it : sample_sequence(model_hypare, 100, batch_size=1,context=”How are you?”)
- use our EmapthicDialog as input data
-