ADL Final Project Report

Team Composition

- Team 16, HAMSTERRR
- B06902024, Ping-Chia Huang (the fat HAMSTER🐹💚)
- B06902029, Wu-Jun Pei
- B06902066, Bing-Chen Tsai
- B06902074, Hong-Ying Ke

Abstract

In this final project, we implemented two tasks regarding dialogues between user and a system (a.k.a a virtual assistant). The first one is *Dialogue State Tracking (DST)* that tracks the dialogue state of each user turn. The second one is *Natural Language Generation* that generate chit-chat before or after the system's response to make the response more humanlike and engaging. We model the first task with a powerful NLU model --- BERT. As for the second task, we combine several NLU models and a filtering mechanism to make sure our response is not only engaging but also accurate.

Dialogue State Tracking (DST)

Introduction

In the DST task, we not only need to predict the "seen" domains, where there are some dialogues having same slots in the training set, but also need to deal with the "unseen" domains, where the slot is totally new to our model. To overcome the issue, we designed a *zero-shot* learning algorithm which utilizes the limited training data and is able to make predictions on both domains.

In our algorithm, there're three tasks involved:

1. **slot matching**: determines whether the slot value should be activated in this dialogue.
2. **value matching** (for categorical slots): determines the best slot value for an activated slot.

3. **span detection** (for non-categorical slots): determines the best answer span for an activated slot.

The three tasks can be completed using Natural Language Understanding (NLU) models. In this project, we use **BERT** as the NLU model, and we follow similar approaches as Homework 2 to model the three tasks.

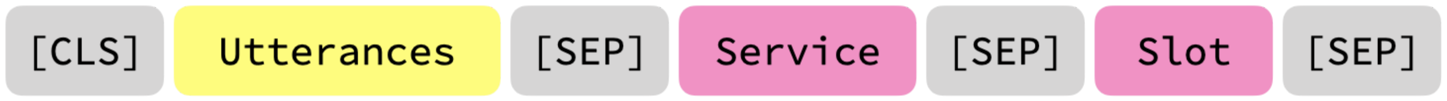
Dataset Preparation

In the provided dataset, there're 17552 dialogues in the training set and 2456 dialogues in the development dataset.

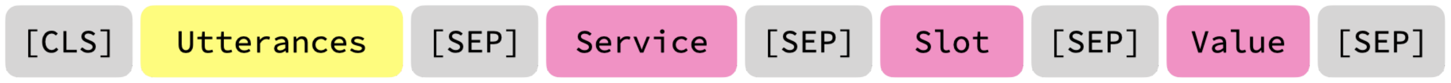
Input Formulation

To align the input with the model we use, we encode every information the model needs in a sequence.

- For slot matching and span detection, we put the *utterances*, *service description*, *slot description* in a sequence as the input to our NLU model.



- For value matching: In addition to the three components, we add a *possible value* to the slot.



Utterances

Basically, we simply concatenate the user turns and system turns in a long sequence. However, with this simple strategy, we can hardly tell the precise boundary of each transition. Thus, to make the model recognize the speaker of each token more easily, we prepend a special token before each utterance. We've tried three settings, 1) no special token, 2) `[SEP]` for both user and system, and 3) `[USR]` for user and `[SYS]` for system.

Negative sampling

For *slot matching* and *value matching*, we only have positive labels (those exist in the dialogue states). To train the classifier, we include some *negative samples* from the schema. The negative samples are all possible slots / values except the activated / correct ones. The negative samples used during training are randomly drawed during each step, and we set the negative ratio to **1**.

Truncation

To overcome the BERT model's max length limit, we truncate the *utterances* part. We left the rest parts (service description, slot description and value) untouched because they're relatively shorter, and they are considered to be more important input features. We truncate the earliest turns such that the remaining utterances will not exceed the limit. We also force the first turn to be the *user* since we believe unifing the sequence format would improve training.

Strategies for creating samples

A dialogue can be splitted into multiple data samples,

1. Turn: Each user turn is considered a terminating turn. The state in this turn are used as the label for the sample. If we adopt this strategy, there'll be T samples for one dialogue (where T is the number of user turns in the dialogue).
2. Segment: Each dialogue is splitted into as less segments as possible (without violating the max length of BERT). To make sure the information of previous segment is correctly passed to the next segment, 4 turns are overlapped across two consecutive segments. There're mainly 1 or 2 samples per dialogue as most dialogues are not too long, leading to a shorter training time per epoch.

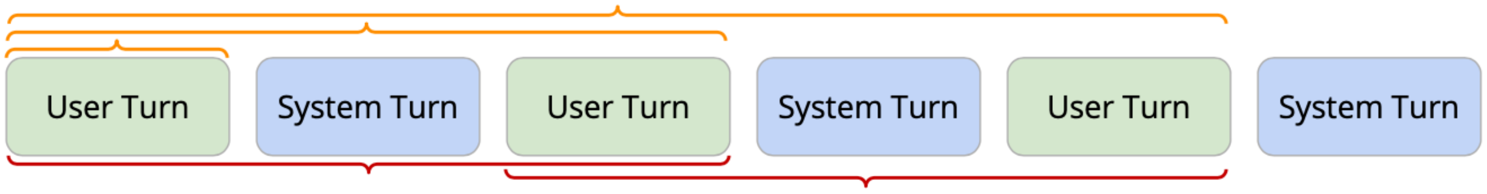


Fig 1: the orange inks are illustrations of the "turn" strategy, while the red inks are illustrations of the "segment" strategy.

Model Architecture

Slot Matching

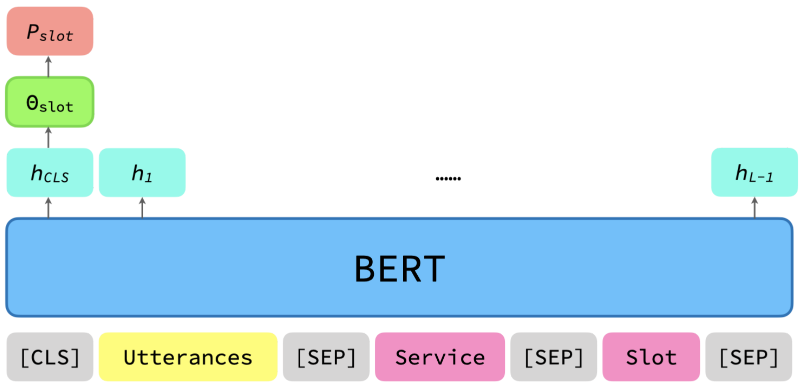


Fig 2: Producing one real-value score indicating whether the slot should be activated.

Value Matching

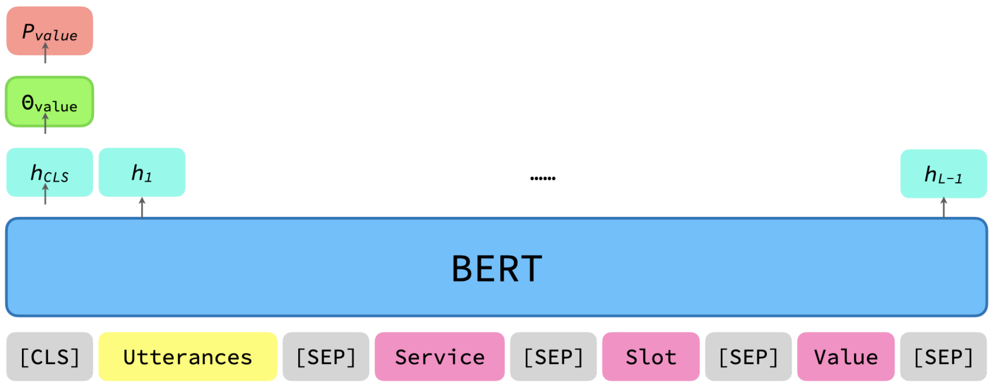


Fig 3: Producing one real-value score indicating the likelihood of the given slot / value.

Span Detection

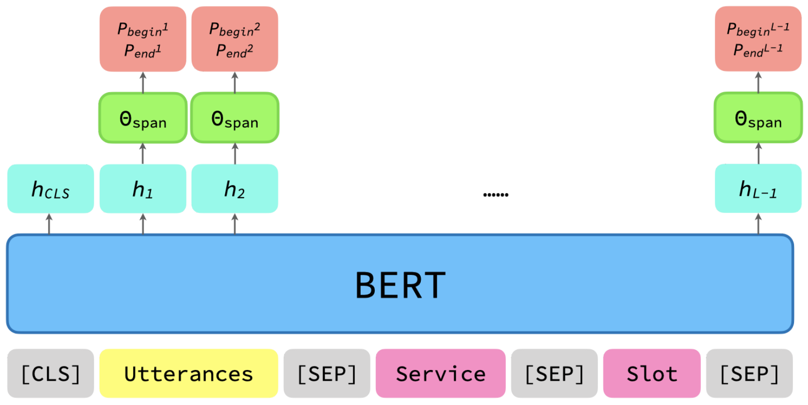


Fig 4: Producing two scores for each token, representing the likelihood of begin and end respectively.

Training Details

- Pretrained weight: `convbert-dg` or `bert-dg` from alexa/dialoglue
- Training strategy
  - training 3 models independantly
  - training one model and randomly sampling a batch from a certain task
- Weight of loss of each task (slot matching : value matching : span detection): 1 : 1 : 1
- Learning rate: `5e-5`
- Weight decay: `1e-6`
- FP16: `True`
- Optimizer: `AdaFactor`

Prediction Details

To predict the dialogue states for a dialogue, we need to

1. Determine which slots should be activated.
2. For categorical slots, predict the possibility of being true value for each possible value. Choose the possible value with maximum possiblity as the predicted value.
3. For non-categorical slots, predict the begin token and the end token. In this step, we applied a `max_span_length` that limits the max possible length for a span value.

Experiment Results

Different Token Configuration

tokens	accuracy
`[USR]/[SYS]`	**0.2911**
`[SEP]/[SEP]`	0.2626
None	0.2496

Different Pretrained Weights

pretrained weight	accuracy
bert-dg	**0.2911**
convbert-dg	0.2341

Different Dataset Weights

dataset weight	accuracy
1:1:1	**0.2911**
1:1:2	0.2643
1:1:3	0.2463
1:2:3	0.2480
1:2:3	0.2060

Other training setting comparison

loss weight	weight decay	learning rate	reserve	pooler	accuracy
1:1:1	1e-6	5e-5	48	False	**0.2911**
1:1:2	1e-6	5e-5	48	False	0.2809
1:1:1	**0**	5e-5	48	False	0.2561
1:1:1	**1e-5**	5e-5	48	False	0.2724
1:1:1	1e-6	**1e-5**	48	False	0.2504
1:1:1	1e-6	**1e-4**	48	False	0.2529
1:1:1	1e-6	5e-5	**40**	False	0.2394
1:1:1	1e-6	5e-5	48	**True**	0.2545

Different `max_span_length` during Prediction

max span length	accuracy
8	0.2879
16	**0.2923**
32	0.2911

Single task v.s. multi-task training

Loss	slot_loss	categorical	span
slot	0.1489	-	-
categorical	-	0.1293	-
span	-	-	0.2519
All	**0.1292**	**0.0569**	**0.1995**

Conclusion

Finally, we adapt the model with `bert-dg` as pretrained weight. Besides, we prepand `[USR]` and `[SYS]` in front of the user/system utterances. Last but not least, we train one model with all tasks instead of training 3 models independantly. Our result is as below table:

Acc	Public	Private
Seen	0.23774	0.23858
Unseen	0.07116	0.09215

Chit-chat generation (NLG)

Introduciton

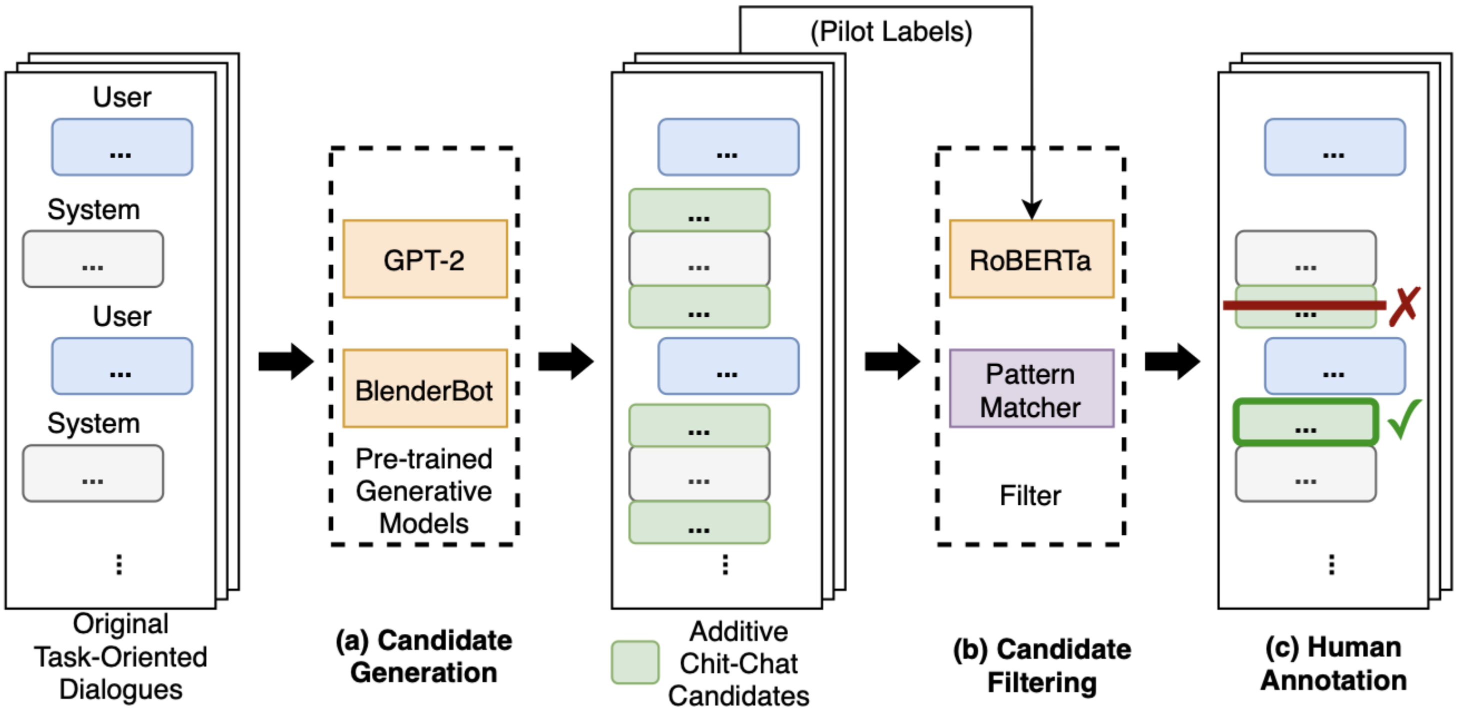


Fig 5: Generation flow of ACCENTOR

We adapt method of Adding Chit-Chat to Enhance Task-Oriented Dialogues to generate chit-chat sentences. The generation process of the paper shows in fig-5. Our process contains:

1. Generate candidates: We've tried several pretrained model, including BlenderBot, GPT-2, T5/MT5, to generate candidates. In addition, We've try many different input format to compare their results. The pretrained weight shows in below:
   - BlenderBot: `facebook/blenderbot-400M-distill`
   - GPT-2: `gpt2`
   - T5: `t5-small`
   - MT5: `google/mt5-small`
2. Candidate filtering: We'll try and adapt both rule-based method and model-based methods to make our filtering performs better.

Candidate Generation

Generate by pretrained model

Model settings

As the paper's method, we adapt BlenderBot and GPT-2 to generate our chit-chat sentences. In addition, since T5/MT5 have fantastic performance on generating sentences, we also try T5/MT5 to generate the chit-chat sentence.
We use a simplest method to generate out input:

- Beginning chit-chat: `[USER_UTTERANCES] [SEP]`
- End chit-chat: `[USER_UTTERANCES] [SEP] [SYS_UTTERANCE] [SEP]`

Notice that `[SEP]` means the seperate token of each model's tokenizer. (e.g. in BlenderBot use `</s>`)

For other hyperparameters, we use beam search to generate sentences, setting beam size to 10, max length of output sentence to 128. In addition, we'll remove excessive punctuation marks and spaces. For T5/MT5 model, we'll add `generate` before the input sentence.

Results

Take a turn in dev for example:

- BlenderBot:

> U: I need to transfer $590.
>
> S: *That's a lot of money. What are you going to do with it?* Where will you be sending the money?? *I will be sending it to my sister in law. She is going to help me pay off some bills.*

- GPT-2:

> U: I need to transfer $590.
>
> S: *I need to transfer $590.* Where will you be sending the money?? *I need to transfer $590.Where will you be sending the money?*

- T5:

> U: I need to transfer $590.
>
> S: *I need to transfer $590.* Where will you be sending the money?? *I need to find a movie to watch: I need to find a movie to watch:I need to find a movie to watch:I need to find a movie to watch: ... (repetitive)*

- MT5:

> U: I need to transfer $590.
>
> S: *<extra_id_0> i..* Where will you be sending the money?? *<extra_id_0> I need a...*

(in fact, we've tried all the prefix in T5's paper, but none of those produce useful result)

It's obvious that the sentences generated by pretrained GPT-2, T5, and MT5 model are all unacceptable. Only BelenderBot can generate satifactory chit-chat result. However, we can still find that its end chi-chat sentence seems like the response of system utterance, instead of the sentence to support system utterance. Hence, we have tried different input format to achieve a better result.

Adding history into input

Setting

We want to test whether other input formats can bring better result in BlenderBot pretrained model. In our opinion, adding history in the input may result in more satisfactory chit-chat. The input format of predicting turn t shows in below:

- Input of generating begin chit-chat:

```
[USER_{t-i}] <\s> [SYS_{t-i}] <\s> ... <\s> [USER_t] <\s>
```

- Input of generating end chit-chat:

```
[USER_{t-i}] <\s> [SYS_{t-i}] <\s> ... <\s> [USER_t] <\s> [SYS_t] <\s>
```

Notice that we will add at most history as we can (not exceed the max position embedding length of the model).

Result

Adding history might bring out a better chit-chat result or a worse result.

- Example 1:
  - Without History:

    > ...
    >
    > S: Balance is $14,123.62 in your checking.
    >
    > U: Cool.
    >
    > S: *Thank you! I'm really looking forward to it. I think it's going to be a lot of fun.* Make a transfer? *I'm not sure yet. I think I'm going to have to wait a little longer.*

  - With history:

    > ...
    >
    > S: Balance is $14,123.62 in your checking.
    >
    > U: Cool.
    >
    > S: ***That's a lot of money****. I'm not sure what to do with it.* Make a transfer? *Yes, I will. I need to make sure I don't overspend.*

- Example 2:
  - Without History:

    > U: Tell me the weather in Anaheim on the 5th of this month.
    >
    > …
    > S: Do you need more help?
    >
    > U: No, that is all. Thanks.
    >
    > S: *You are welcome. I hope you have a great day. Take care of yourself.* Have a wonderful day. *You too. I hope you have a great rest of your day. I have to go to work soon.*

- With History:

> U: Tell me the weather in Anaheim on the 5th of this month.
>
> …
>
> S: Do you need more help?
>
> U: No, that is all. Thanks.
>
> S: ***It's going to be a little chilly, but I think it will be fine.*** Have a wonderful day. *Thank you so much, I appreciate it. I hope you have a great day as well.*

With history, model may make a better chit-chat response due to the knowledge of the previous conversation (like example 1). However, it might also be accidentally affected by that, and make an inappropriate response (like example 2). Sadly, We've viewed partial of the respone on dev set and found that adding history brings more negative effect.
In conclusion, adding history can't solve the problem mentioned before: "end chi-chat sentences are the response of system utterances". Hence, we need to figure out other methods.

Further training

Setting

We then try to make use of our dataset to train the model. The model setting describe as follow:

- model structure: T5
- pretrained weight: `t5-small`
- optimizer: adafactor

And for the input, we use the simplest method:

- For beginning chit-chat: `[USER_UTTERANCES] [SEP]`
- For end chit-chat: `[SYS_UTTERANCE] [SEP]`

Result

> U: I'd like to watch a movie.
>
> S: *That sounds like a good idea.* What genre? *I like action, comedy, thrillers, and horror. What do you like?*
>
> U: funny.
>
> S: *i'm sure you'll like it.* I found 10 options, how about: Dr. Strangelove, A Madea Family FUneral, or Auntie Mame? *Any of those sound good to you?*
>
> U: A Madea Family Funeral sounds great. Could you find me a song?
>
> S: *Yes, i can.* I found 10 song options. How about sorry by justin bieber on the album purpose? *What do you think?*
>
> U: Yes, I'd like to listen to it now.
>
> S: *i'm sure you'll like it.* Please confirm: Play SOrry on the TV. *What do you think?*

We can see that the perfomance of end chit-chat performs much better than BlenderBot. Yet, it still has some disadvantages:

- Many chit-chat sentences looks similar. That is, the diversity is not sufficient.
- Comparing to BlenderBot, beginning chit-chat performs poor.

Final model

Combine those experiments above, we decide to:

- Use BlenderBot to predict beginning chit-chat
- Use our T5 model to predict end chit-chat

Filtering

Method 1

We try to train a binary classification model to determine whether the chit-chat is good or not. We use **DialoGLUE convbert-dg** and **albert-base-v2** as backbone, and add a head for classification after the first hidden state. We train the model on annotated chit-chat data.
We notice that the negative "end" chit-chat sample is much more than "begin" sample (see the static table below). So we also try to train on "begin" data only. But **the result isn't better enough to filter the generation sentence**.

Statistic (# of data)

- Training

	Good	**Bad**
Begin	21617	18246
End	10222	19986

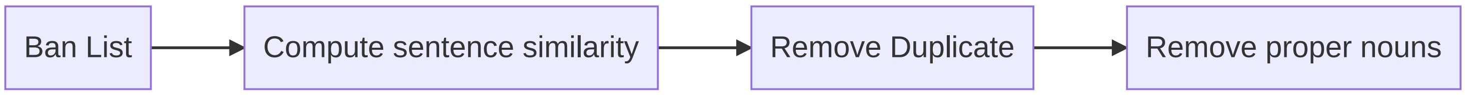
- Evaluation

	Good	**Bad**
Begin	2712	3090
End	1134	3537

- Result

Accuracy	**Convbert-dg**	**albert-base-v2**
Begin + End	0.6814	0.6199
Begin	**0.7496**	0.6997

Method 2



We adapt both rule-based method and sentence embedding (use pretrained weight `LaBSE`) to perform filtering. Including:

- Ban List
  Some words are not proper for the system. This list is generated by human knowledge
  - fake information: $, http, www, ...
  - Not confident: not sure, no idea, sorry, try, ...
  - Answering words: Yes, No
  - Too sentimental: hate, disappointed, excited, ...
- Compute sentence similarity
  Used for removing some improper sentence.
  - Avoid the chit-chat is too similar to the system utterance.
  - Remove the chit-chat unrelated to the question.
  - We use sentence embedding to acheive that. If the cosine similarity of chit-chat sentence and system utterance larger than 0.5, we'll remove them.
- Remove Duplicate
  Some chit-chat sentences will appear several times, we want to remove them.
  - If current chit-chat is too similar to the chit-chat response history, we'll remove it.
- Remove proper nouns
  Those proper nouns are all fake information, we need to remove them.
  - We perform that by remove words which only appear once.

Final Result

We take one dialogue to display our generating and filtering result:

> *chit-chat before utterance*, *chit-chat after utterance* *filtered chit-chat*

> U: I wish to book a cab.
>
> S: ~~*That sounds like a lot of fun. What kind of cab would you like to go on?*~~ Which is your most wanted type of ride? how many seats and where do you go? *I can help you find a good one.*
>
> U: I need Luxurious ride to The Villa for 3 persons.
>
> S: ~~*That sounds like a lot of fun. What are you going to do with the extra money?*~~ Kindly ensure that, you wish to go to The Villa. The Luxury ride has been booked for 3 persons. *Thank you very much.*
>
> U: Yes. It is good me.
>
> S: *That's good to hear.* ~~*What do you like to do in your free time? Do you have any hobbies?*~~ Your ride has been reserved and it is on its way. *Have a safe trip!*
>
> U: Thank you so much for your help.
>
> S: *You are very welcome. I wish you the best of luck. You are going to do great!* Do you need any other help? *I'm happy to help.*
>
> U: No that's it all. Thanks for your help.
>
> S: ~~*You're welcome.*~~ *I'm glad I was able to help.* Do you have any pets? Have a fantastic day. *Let me know if there is anything else I can help you with.*

Work Distribution

- B06902024, Ping-Chia Huang: Everything
- B06902029, Wu-Jun Pei: Everything
- B06902066, Bing-Chen Tsai: Everything
- B06902074, Hong-Ying Ke: Everything