

Football Dialogue Assistant

Anh Tu Phan - 1st year student
Master of Computer Science
anhtu.phan@studenti.unitn.it

June 2021

In this project, I will try to build a dialogue system that will provide the information about football such as the league fixtures, team information, player statistic, ... The following is an example of conversation between user (label as **U**) and the system (label as **B**)

U: Hi
B: Hello, How can I help you?
U: Tell me about Euro
B: There are 4 matches coming up:
 In 8th Finals:
 Croatia meet Spain on Jun 28 06:00PM;
 France meet Switzerland on Jun 28 09:00PM;
 England meet Germany on Jun 29 06:00PM;
 Sweden meet Ukraine on Jun 29 09:00PM;
U: top 2 players of the Euro now?
B: The following is 2 top player:
 1: Cristiano Ronaldo dos Santos Aveiro of Portugal with 5 goals.
 2: Romelu Lukaku Menama of Belgium with 3 goals.
U: How many goals did Ronaldo score at Euro?
B: At Euro Championship: total is 5; conceded is 0; assists is 1;
U: The next matches of England at Euro?
B: England will meet Germany at 8th Final of Euro Championship
 in Jun 29 06:00PM at home
U: The position of England?
B: England is participating UEFA Nations League,
 World Cup - Qualification Europe, and Euro Championship.
 Which league do you want to know about?
U: Euro Championship
B: In Euro Championship, England is at 1 position with 7 point

System Architecture

The Figure 1 show the architecture of dialogue system. There are three main components in the system:

- **Google Assistant** will interact with user. It converts user's speech to text (Automatic Speech Recognition) and converts text generated from **Rasa** to speech to respond to user (Text-to-Speech Synthesis).
- **Rasa** is main core component of system. It will generate information base on user's text input.
 - Spoken Language Understanding will receive text message and then extract the *intent* and *entities* from message
 - Dialogue Management will keep track of conversation *state* (stores concept values (entities and actions) into slots). It decide the next action based on state and user input (intent and entities) and update states (provide information to fill response). In this project, it will request Football News component to fill the information that user ask.

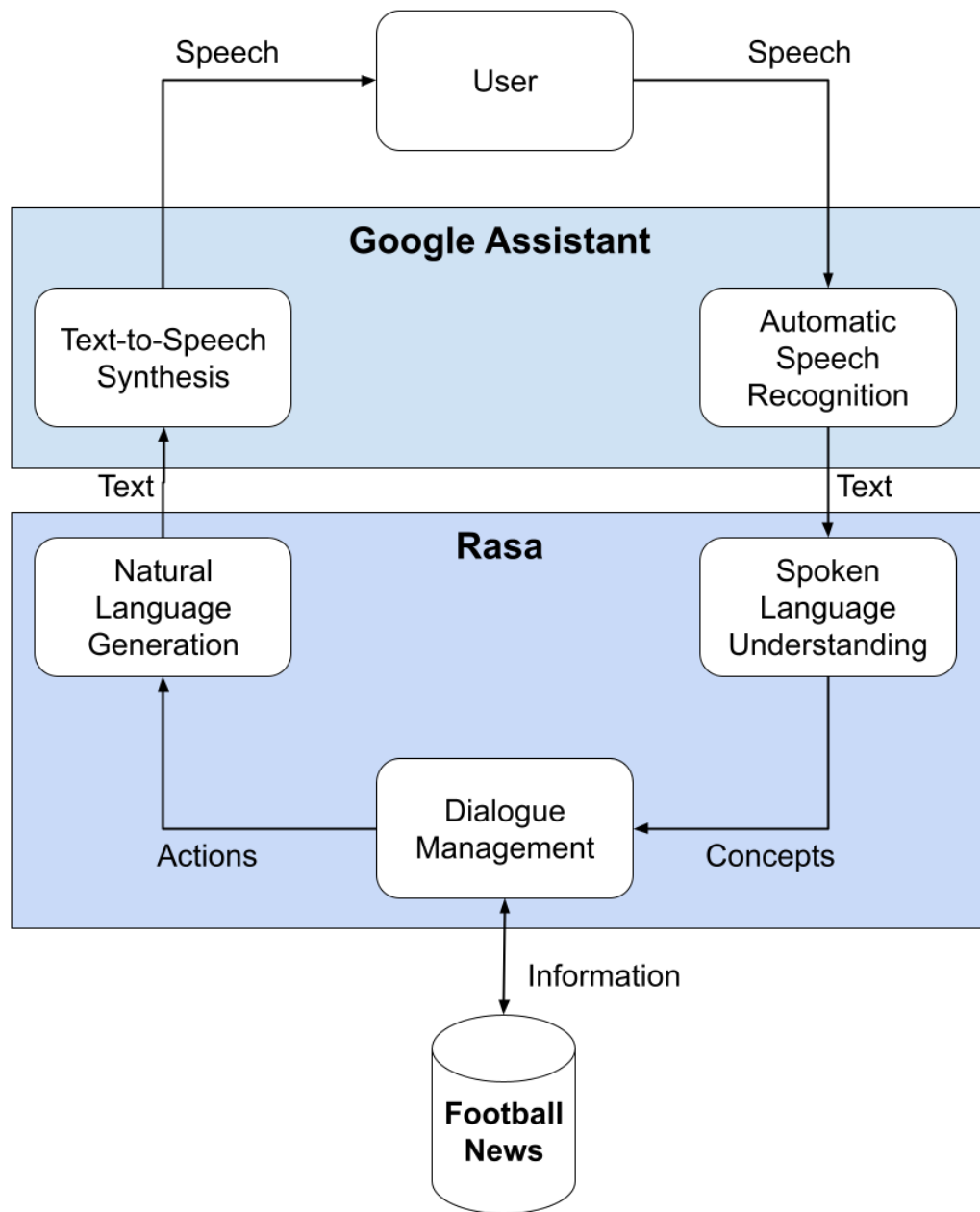


Figure 1: System Architecture

- Natural Language Generation will generate message to response to user.
- **Football News** will provide information such as league fixtures, player statistic, team information, ... to Dialogue Management. In this project, I will extract these information from API-FOOTBALL [1].

Implementation

Intents and Entities

To handle request from user, we need to classify the request to intents to understand type of request and then extract entities from intent to response suitable information. In this project, I will define six intents and five entities to handle utterance of user. Table 1 show the information of these intents and entities could be extracted from them.

Intent	Meaning	Entities	Examples
<code>ask_league_info</code>	request information of one specify league	<code>league_name</code>	Tell me about [Euro] (<code>league_name</code>)
<code>ask_top_player</code>	request information about the players who score the most goals in one specify league	<code>league_name</code> , <code>query_number</code>	top [2] (<code>query_number</code>) player of [Euro] (<code>league_name</code>)
<code>ask_player_info</code>	request information about statistic of player such as number of goals, assists, ...	<code>query_type</code> , <code>PERSON</code> , <code>league_name</code>	number [goals] (<code>query_type</code>) of [Kai Havert] (<code>PERSON</code>) at [euro] (<code>league_name</code>)
<code>ask_fixtures</code>	request information about fixtures of a club at one or all participated leagues	<code>club_name</code> , <code>league_name</code>	the fixtures of [england] (<code>club_name</code>) at [euro] (<code>league_name</code>)
<code>ask_club_info</code>	request the position of a club at one or all participated leagues	<code>club_name</code> , <code>league_name</code>	update position of [england] (<code>club_name</code>) at [euro] (<code>league_name</code>)
<code>provide_league_info</code>	specify which league user want to know the position of a club	<code>league_name</code>	[euro] (<code>league_name</code>)

Table 1: Intent and entities

There are six corresponding custom actions to handle six intents classified from user's utterance. These actions will get extracted information from API-FOOTBALL that suitable with entities extracted from user's utterance. In detail:

- `action_provide_league_info` will handle intent `ask_league_info`. This action will get the fixtures of current round of entity `league_name` to response to user.
- `action_top_player` will handle intent `ask_top_player`. It will get top `query_number` scorers at `league_name` to return to user. If entity `query_number` is not provided, this action will return the information of the highest scorer.
- `action_player_info` will handle intent `ask_player_info`. It will get name of player from entity `PERSON` and return the `query_type` (goals, assists) statistic at `league_name` to user if it is specified otherwise return the information of `query_type` in all leagues that `PERSON` participates.
- `action_fixtures` will handle intent `ask_fixtures`. It will provide the next 5 matches of `club_name` at `league_name` if it is provided otherwise return the next 5 matches coming up at all league participated by `club_name`.

- `action_club_info` will handle intent `ask_club_info`. If user ask information in specified league which mean `league_name` entity is provided, this action will return the position of `club_name` at `league_name`. Otherwise, if `league_name` is not provided, this action will provide all leagues that `club_name` is participating and ask user chose one `league_name` or `all`. Then the action `action_club_info_spec` will provide the position information of `club_name` at `league_name` which is specified by user through `provide_league_info` intent.

Pipeline Configuration

Because `league_name`, `club_name`, `query_type` have known values, for example `league_name` may have values: Premier League, Euro Championship, ... `club_name` may have values: Chelsea, England, ... `query_type` is chosen from: goals, assists. I plan to use *lookup tables* from `RegexEntityExtractor` for extracting these entities. But in [Rasa documentation](#), they recommend not use this extractor in combination with `MitieEntityExtractor`, in addition I cannot find the way to specify entity types will be applied by determined extractor. So, I will use `RegexFeaturizer` to use regexes matches (*lookup tables*) as additional signal for statistical extractor. Moreover `RegexFeaturizer` for entity extraction are currently only supported by the `CRFEntityExtractor` and the `DIETClassifier` components [2].

The `player_name` is a name of players. Therefore, to take advantage from pre-trained model, I will use dimension `PERSON` of `SpacyEntityExtractor` to extract `player_name` entity from user's utterance.

The `query_number` entity may have the different type such as a word four instead of a number 4. Therefore, I will use `EntitySynonymMapper` to recognize these cases.

Base on above observations, I will try three different pipeline configuration:

1. Using MITIE for intent classifier and entity extractor (which is called `config_v1`)

```
- name: MitieNLP
  model: "data/total_word_feature_extractor.dat"
- name: MitieTokenizer
- name: MitieFeaturizer
  pooling: "mean"
- name: LexicalSyntacticFeaturizer
- name: MitieIntentClassifier
- name: MitieEntityExtractor
- name: EntitySynonymMapper
```

2. Using `DIETClassifier` for intent classifier and combine `SpacyEntityExtractor` with `DIETClassifier` for entity extractor (which is called `config_v2`)

```
- name: SpacyNLP
  model: es_core_news_md
- name: SpacyTokenizer
- name: SpacyFeaturizer
- name: LexicalSyntacticFeaturizer
- name: DIETClassifier
  batch_strategy: balance
  intent_classification: True
  entity_recognition: False
  epochs: 500
  constrain_similarities: true
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
- name: SpacyEntityExtractor
  dimensions: ["PERSON"]
- name: RegexFeaturizer
- name: DIETClassifier
  batch_strategy: balance
```

```

        intent_classification: False
        entity_recognition: True
        epochs: 100
- name: EntitySynonymMapper

```

3. Using MITIE for intent classifier and combine `SpacyEntityExtractor` with `DIETClassifier` for entity extractor (which is called `config_v3`)

```

- name: MitieNLP
  model: "data/total_word_feature_extractor.dat"
- name: SpacyNLP
  model: es_core_news_md
- name: MitieTokenizer
- name: MitieFeaturizer
  pooling: "mean"
- name: LexicalSyntacticFeaturizer
- name: MitieIntentClassifier
- name: SpacyEntityExtractor
  dimensions: ["PERSON"]
- name: RegexFeaturizer
- name: DIETClassifier
  batch_strategy: balance
  intent_classification: False
  entity_recognition: True
  epochs: 100
- name: EntitySynonymMapper

```

To evaluate of pipeline configurations, I will run each configuration above two times with different percentages of training data. The full comparison of results is found [here](#). When running comparison in full dataset, the Figure 2 show the intent confusion matrix between `config_v1` and `config_v2`, intent confusion matrix between `config_v2` and `config_v3` is shown in Figure 3. The entity extractor confusion matrix comparison between `config_v1` and `config_v2` is shown in Figure 4 and between `config_v2` and `config_v3` is shown in Figure 5.

From intent confusion matrix comparison between `config_v1` and `config_v2` (Figure 2), it is shown that the better result is achieved when apply MITIE for intent classification compare with `DIETClassifier`. For example, while all `ask_fixtures` intent is truly predicted in MITIE classifier, one utterance with `ask_fixtures` intent is predicted as `ask_player_info` in `DIETClassifier`. The wrong prediction also happen between `ask_league_info` and `ask_fixtures` in `DIETClassifier`.

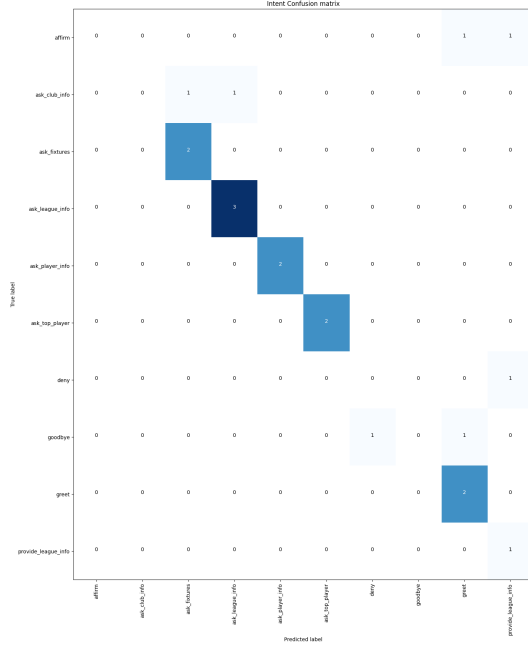
From entity extractor confusion matrix comparison between `config_v1` and `config_v2` (Figure 4), because of take advantage of pre-trained `SpacyEntityExtractor` in PERSON entities, the `config_v2` predicts truly while `config_v1` mistakenly predict with `league_name` and `no_entity`.

Base on two observations above, I will chose `config_v3` which is combined the advantage of MITIE for intent classifier and `SpacyEntityExtractor` for PERSON entity extractor, the other entities may be extracted from `DIETClassifier`. As shown in Figure 3 and Figure 5, the result of `config_v3` will better than `config_v2` in intent classifier because it use MITIE instead of `DIETClassifier`; and the result in entities extractor will be same because they both use `SpacyEntityExtractor` and `DIETClassifier`.

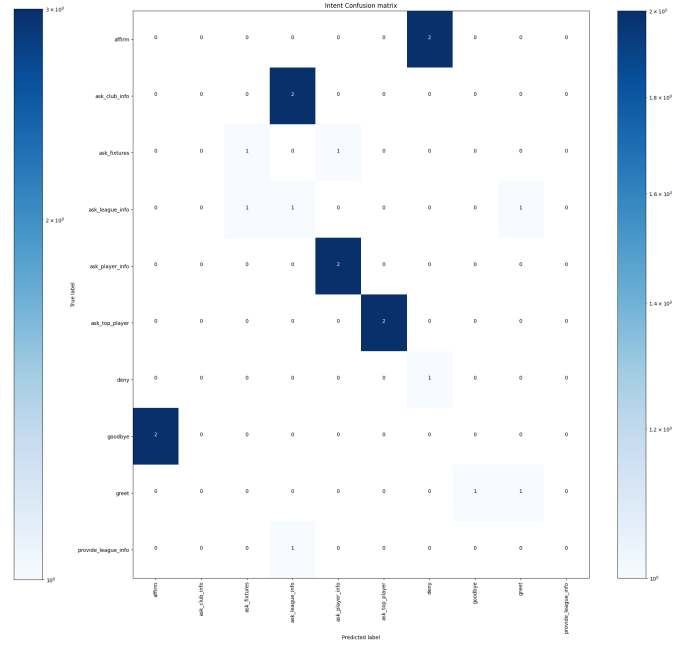
Conclusion

In this project, I have construct a simple dialogue system to provide some information about football. A system probably work but still simple. The number intent is limited (6 intent) so can serve limit request from user. Because of having to build data from scratch, the number of utterance for training is small.

For future work, I will build or find more data to train better model; build more intents to provide more information such as line-up, information related to a match such as confrontation history; build more complicated information such as instead of just provide position also provide the different point to

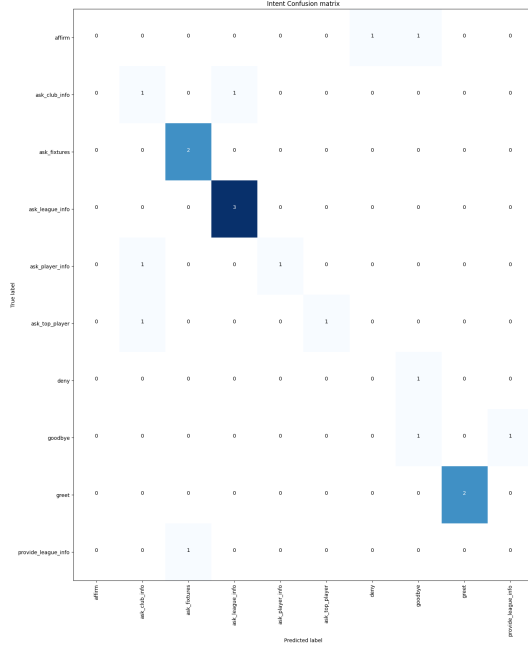


(a) Intent confusion matrix of **config_v1**

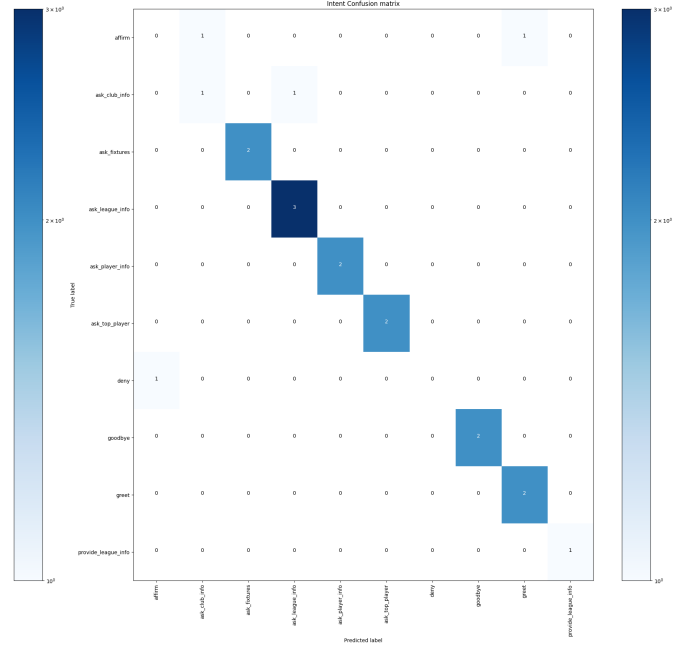


(b) Intent confusion matrix of **config_v2**

Figure 2: Intent confusion matrix comparison between **config_v1** and **config_v2**

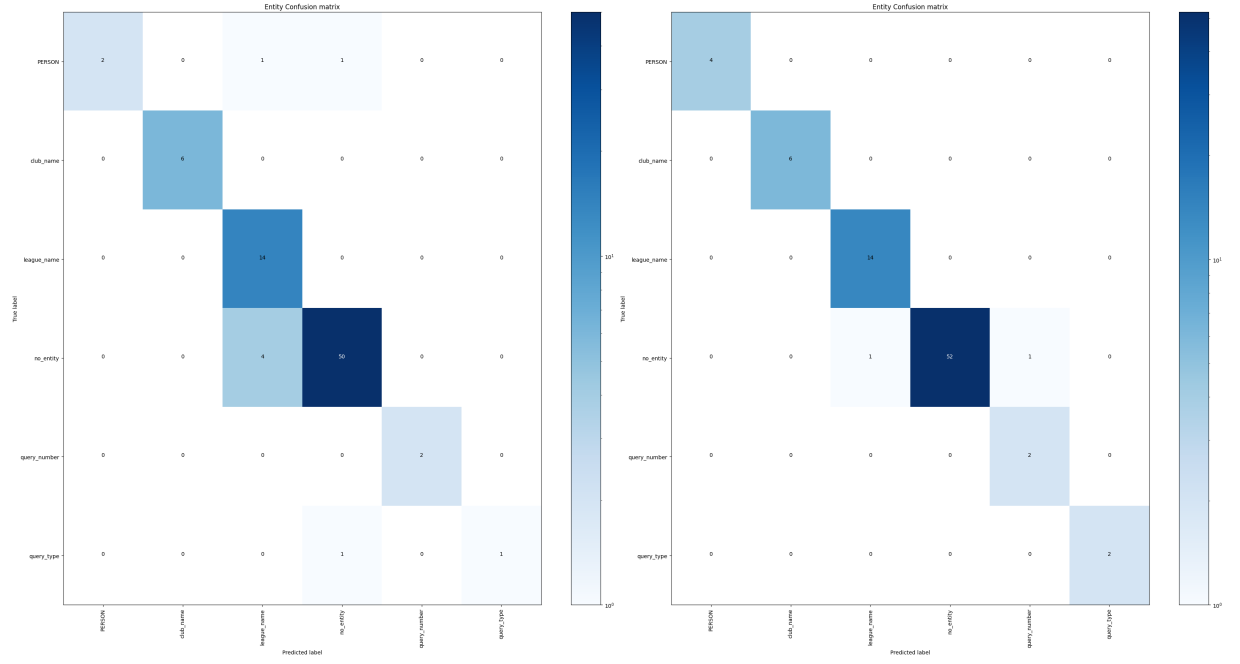


(a) Intent confusion matrix of **config_v2**



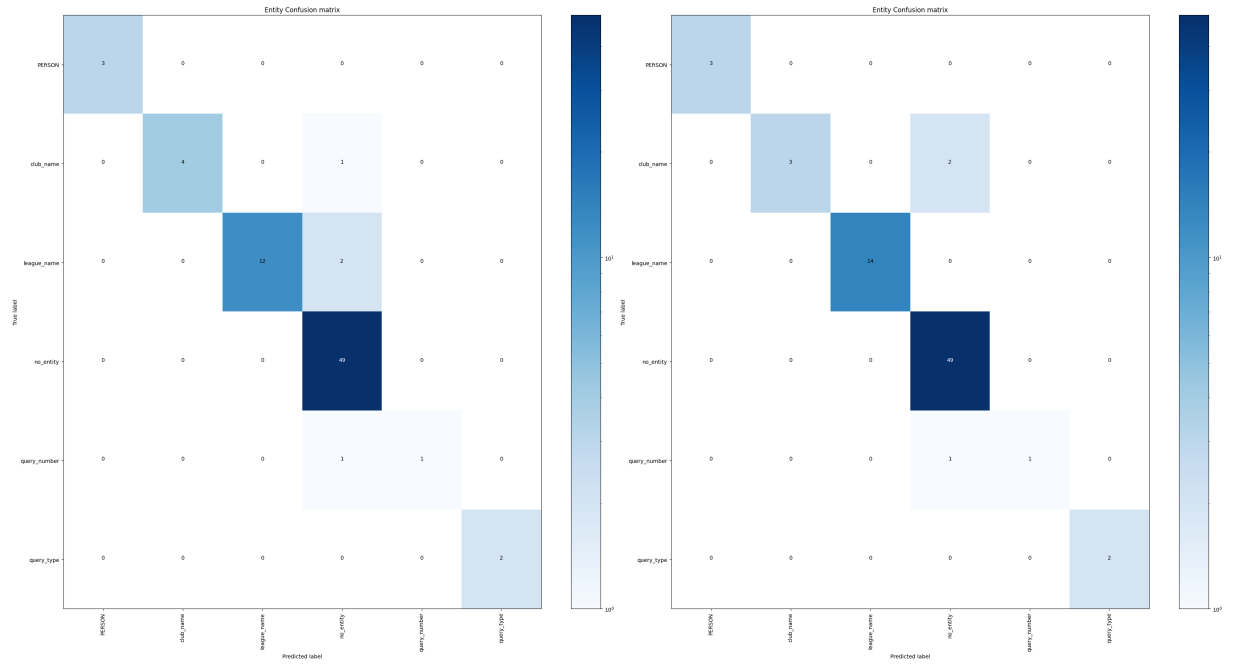
(b) Intent confusion matrix of **config_v3**

Figure 3: Intent confusion matrix comparison between **config_v2** and **config_v3**



(a) Entity Extractor confusion matrix of `config_v1` (b) Entity Extractor confusion matrix of `config_v2`

Figure 4: Entity Extractor confusion matrix comparison between `config_v1` and `config_v2`



(a) Entity Extractor confusion matrix of `config_v2` (b) Entity Extractor confusion matrix of `config_v3`

Figure 5: Entity Extractor confusion matrix comparison between `config_v2` and `config_v3`

achieve a important position such as champion, top-4 or relegation position; and build more "intelligent" bot such as automatically summary related information that user request from newspaper.

References

- [1] Api-football, <https://rapidapi.com/api-sports/api/api-football/>.
- [2] Rasa regex featurizer documentation, <https://rasa.com/docs/rasa/components#regexfeaturizer>.
- [3] Language understading systems lab, <https://github.com/esrel/LUS>.
- [4] Rasa documentation, <https://rasa.com/docs/rasa/>.