

# **Information Extraction Documenta- tion**

***Release***

**FB Persona**

July 24, 2018



|          |                                       |               |
|----------|---------------------------------------|---------------|
| <b>1</b> | <b>IEApp</b>                          | <b>1</b>      |
| 1.1      | driver module . . . . .               | 1             |
| 1.2      | database module . . . . .             | 1             |
| 1.3      | filemanager module . . . . .          | 2             |
| 1.4      | clean module . . . . .                | 2             |
| 1.5      | emojiremove module . . . . .          | 3             |
| 1.6      | entity module. . . . .                | 4             |
| 1.7      | spellcheck module . . . . .           | 4             |
| 1.8      | alchemyapi module . . . . .           | 5             |
| 1.9      | asstypes module . . . . .             | 6             |
| 1.10     | conceptnet module. . . . .            | 7             |
| 1.11     | contextunderstanding module . . . . . | 7             |
| 1.12     | dbpedia module . . . . .              | 8             |
| 1.13     | fanextract module . . . . .           | 8             |
| 1.14     | foodextract module. . . . .           | 10            |
| 1.15     | gameextract module . . . . .          | 11            |
| 1.16     | googlesearch module. . . . .          | 13            |
| 1.17     | likedpagesevents module . . . . .     | 14            |
| 1.18     | sportextract module . . . . .         | 16            |
| 1.19     | translator module. . . . .            | 17            |
|          | <br><b>Python Module Index</b>        | <br><b>19</b> |
|          | <br><b>Index</b>                      | <br><b>21</b> |



## 1.1 driver module

**class** `driver.Driver`

Bases: `object`

**init** ( *name, data* )

Main function of the Information Extraction process

**Parameters**

- **persona** – persona of the user
- **name** – Facebook name of the user
- **data** – type of data to be extracted (1 for posts, 2 for liked pages, and 3 for events)

**Returns**

## 1.2 database module

**class** `database.Database`

Bases: `object`

**connectDB** ( )

Connects to the Database

**Returns** cursor in the database

**getDateTime** ( *timestamp* )

Converts the timestamp to date and time

**Parameters** **timestamp** – date and time the post was created

**Returns** date, time

**getEventsByLabel** ( *persona, tableName* )

Gets the events by a specific user preference

**Parameters**

- **persona** – user preference
- **tableName** – table name of the user

**Returns** array of events

**getLikesByLabel** ( *persona*, *tableName* )

Gets the liked pages by a specific user preference

**Parameters**

- **persona** – user preference
- **tableName** – table name of the user

**Returns** array of liked pages

**getPostByLabel** ( *persona*, *tableName* )

Gets the posts by a specific user preference

**Parameters**

- **persona** – user preference
- **tableName** – table name of the user

**Returns** array of posts

**getProfile** ( *name* )

Gets the profile information of the user from the database

**Parameters** **name** – Facebook name of the user

**Returns** instance of a profile

## 1.3 filemanager module

**class** filemanager.**FileManager**

Bases: `object`

**writeFile** ( *filename* )

Writes file (only applicable if exporting the results to .txt) :param filename: desired name of the file

**Returns**

## 1.4 clean module

**class** clean.**Clean**

Bases: `object`

**cleanData** ( *posts*, *profile* )

Main function for cleaning the data

**param posts** list of raw posts

**param profile** profile object of the user

**Returns** array of cleaned posts

**dividePost** ( *post* )

Dividing the raw post to 3 main elements

**param post** raw post

**Returns** event, shared post caption, user thought

**extractElements ( *p* )**

Extracts the hashtags and mentions from the post

**param p** an instance of a post (with no values for hashtags and mentions)

**Returns** an instance of a post (with values for hashtags and mentions)

**mild\_clean ( *query* )**

Function for mild cleaning

**param query** raw post

**Returns** cleaned post

**thorough\_clean ( *post* )**

Function for thorough cleaning

**param post** raw text

**Returns** cleaned text

## 1.5 emojiremove module

**class emojiremove.EmojiRemove**

Bases: `object`

A class for removing not only the emoji/emoticons, but also other entities like new lines, whitespaces, URLs and email addresses.

**removeExtras ( )**

Removes the URLs and Email Addresses in the post

**param text** raw text

**Returns** cleaned text

**removeNewLine ( )**

Removes the new lines in the post

**param text** raw text

**Returns** cleaned text

**removeWhitespaces ( )**

Removes the whitespaces in the post

**param text** raw text

**Returns** cleaned text

**remove\_emoji ( )**

Removes the emojis/emoticons in the post

**param text** raw text

**Returns** cleaned text

**remove\_specialChars ( )**

Removes the special characters (except `!?,'"@&:/-]+`) in the post

**param text** raw text

**Returns** cleaned text

## 1.6 entity module

**class** `entity.Entity`

Bases: `object`

**doesExists** ( *token* )

Determines whether a certain word or entity exists

**param token** specific token

**Returns** True or False

**isAnEntity** ( *token* )

Determines whether the token is an entity or not

**param token** specific token

**Returns**

## 1.7 spellcheck module

**class** `spellcheck.SpellChecker`

Bases: `object`

**compareWord** ( *word, fil, eng* )

Compares the original text to the suggested filipino and english text

**param word** original text

**param fil** Suggested Filipino text

**param eng** Suggested English text

**Returns** Filipino or English word (which one is closest)

**findMostSimilar** ( *word, tokens* )

Finds the most similar word from the list of suggested spelled words

**param word** original text

**param tokens** array of suggested spelled words from HunSpell

**Returns** most similar text

**inPeople** ( *text, people* )

Checks if the token belongs to the mentioned people

**param text** text

**param people** array of mentioned people

**Returns** True or False

**spell** ( *text, post* )

Main function for spell corrector

**param text** post that has already undergone cleaning process beforehand

**param post** raw post

**Returns** spelled post



**spellCorrect** ( *token* )

Distinguishes whether the word is spelled correctly or not.

**param token** original text

**Returns** True or False

**spell\_Eng** ( *token* )

Determines whether the spelling is in the English dictionary and gets the suggested English words

**param token** original text

**Returns** array of suggested spelled English words from HunSpell

**spell\_Fil** ( *token* )

Determines whether the spelling is in the Filipino dictionary and gets the suggested Filipino words

**param token** original text

**Returns** array of suggested spelled Filipino words from HunSpell

## 1.8 alchemyapi module

**class** alchemyapi.**AlchemyAPI**

Bases: `object`

**data** = None

**getCategories** ( *query* )

Get the value from the categories feature of Watson API with  $\geq 75\%$  relevance according to category type

**param query** category type

**Returns** JSON of categories

**getCategoriesNoRelevance** ( )

Get the value from the categories feature of Watson API with no relevance

**Returns** JSON of categories

**getCategoriesNoRelevance2** ( )

Get the 3rd level hierarchy from the categories feature of Watson API

**Returns** array of 3rd level hierarchies

**getConcepts** ( )

Get the value from the concepts feature of Watson API with no relevance

**Returns** JSON of concepts

**getConceptsRelevance** ( )

Get the value from the concepts feature of Watson API with  $> 85\%$  relevance

**Returns** JSON of concepts

**getEntities** ( )

Get the value from the entity feature of Watson API

Returns JSON of entities

**getKeywords ( )**

Get the value from the keywords feature of Watson API with > 50% relevance

Returns JSON of keywords

**getKeywordsNoRelevance ( )**

Get the value from the keywords feature of Watson API with no relevance

Returns JSON of keywords

**getRelations ( )**

Get the value from the relations feature of Watson API

Returns JSON of relations

**getSemanticRoles ( )**

Get the value from the semantic role feature of Watson API

Returns JSON of semantic roles

**getSentiment ( )**

Get the value from the sentiment feature of Watson API

Returns sentiment

**natural\_language\_understanding = None**

**nlu ( *query* )**

API Call to IBM Watson API to get the information about the query  
**param query**

Returns True (if without error/s) or False (if with error/s)

## 1.9 asstypes module

**class asstypes.AssTypeInfo**

Bases: `object`

**fangirlboy ( *data, assertions* )**

Identifying assertion type for The Fangirl/Fanboy

**param assertions** array of extracted information

**param data** array of data frames

Returns array of The Fangirl/Fanboy dataframes

**foodie ( *data, assertions* )**

Identifying assertion type for The Foodie

**param assertions** array of extracted information

**param data** array of data frames

Returns array of The Foodie dataframes

**getAssertion** ( *persona*, *data* )

Gets the data frames with identified assertion types

**param persona** persona

**param data** array of data frames

Returns list of data frames with identified assertion types

**getChecklist** ( *persona* )

Gets the list of information to be extracted for each posts

**param persona** persona

Returns JSON of information

## 1.10 conceptnet module

**class** conceptnet . **ConceptNet**

Bases: `object`

**getIsTypeOf** ( *query* )

Get the “Is Type Of” of the query

**param query** query

Returns types

**search** ( *query* )

Searches through ConceptNet

**param query** query

Returns edges

## 1.11 contextunderstanding module

**class** contextunderstanding . **ContextUnderstanding**

Bases: `object`

**asstypeinfo** = <context.asstypes.AssTypeInfo object>

**fan** = <context.fanextract.FanExtractor object>

**foodie** = <context.foodextract.FoodieExtractor object>

**game** = <context.gameextract.GameExtractor object>

**getContext** ( *persona*, *cleaned*, *profile* )

Function for context understanding

**param persona** specific user preference of the user

**param cleaned** list of cleaned posts

**param profile** profile object of the user

Returns array of data frames with assertion types

**sport** = <context.sportextract.SportExtractor object>

## 1.12 dbpedia module

**class** `dbpedia.DBPedia`

Bases: `object`

**getHypernym** (*query*)

Gets the hypernym of the query

**param query** query

Returns hypernym

**getType** (*query*)

Gets the type of the query

**param query** query

Returns array of types

**nlp** = <spacy.lang.en.English object>

**search** (*query*)

Distinguishes whether the query exists on DBPedia

**param query** query

Returns True or False

**searchFoodCategory** (*query*)

Gets the category of food

**param query** food

Returns array of food categories

**sparql** = None

## 1.13 fanextract module

**class** `fanextract.FanExtractor`

Bases: `object`

Context Understanding for The Fangirl/Fanboy

**createAssertionStoryPost** (*base\_data, post, postInfo, storyInfo, assertions*)

extracts information from event and shared post caption

**param base\_data** data frame that contains the specific information that can be extracted

**param post** post instance

**param postInfo** shared post caption

**param storyInfo** event

**param assertions** list of already extracted information

Returns list of extracted information

**describeSubject** ( *sentiment* )

Extracts data from “describing the subject” subject situation

**param sentiment** user thought

**Returns** describe data

**extractPost** ( *post* )

Extracts data from the shared post caption

**param post** shared post caption

**Returns** post data

**extractSentiment** ( *sentiment* )

extracts information from the user thoughts

**param sentiment** user thought

**Returns** activity data, describe data, subject data, sentiment data

**extractStory** ( *story* )

Extracts data from the event

**param story** event

**Returns** event data

**fanExtract** ( *post, persona, person* )

Main function for The Fangirl/Fanboy context understanding

**param post** specific post

**param persona** persona

**param person** user

**Returns** array of extracted information

**findSubject** ( *post* )

Finds the subject of the post

**param post** post

**Returns** post data

**findTitleInQuotes** ( *post* )

Extracts the title of an object that are enclosed with “”

**param post** post

**Returns** title

**find\_adj** ( *docu* )**find\_verb** ( *docu* )**getSentiment** ( *sentiment* )

Gets the sentiment from Watson API

**param sentiment** user thought

**Returns** sentiment

**isArt** ( *title* )

Determines whether the entity is a novel, book, song, or movie

**param title** title of the entity

**Returns** True or False

**narratingActivity** ( *sentiment* )

Extracts data from narrating an activity subject situation

**param sentiment** user thought

**Returns** activity data

## 1.14 foodextract module

**class** foodextract.FoodieExtractor

Bases: `object`

Context Understanding for The Fangirl/Fanboy

**api** = <context.alchemyapi.AlchemyAPI object>

**asstypeinfo** = <context.asstypes.AssTypeInfo object>

**conceptnet** = <context.conceptnet.ConceptNet object>

**dbpedia** = <context.dbpedia.DBPedia object>

**extractFoodPostInformation** ( *post* )

extracts the food (subject) from the post

**param post** post

**Returns** food, food organization

**extractStoryInformation** ( *story* )

extracts information from the event

**param story** event

**Returns** action, organization, location, tagged\_friends

**find\_root** ( *docu* )

finds the root token of a sentence

**param docu** sentence/query

**Returns** root token

**foodieExtract** ( *post, persona, person* )

Main function for The Foodie context understanding

**param post** specific post

**param persona** persona

**param person** user

**Returns** array of extracted information

**getFood** ( *keywords* )

determines if the keywords generated from Watson API has a food entity

**param keywords** array of keywords

**Returns** array of foods

```

getFoodType ( query )
    distinguishes the food type
    param query food
    Returns array of categories

getPersonMentioned ( )
    get the mentioned name of a person
    Returns array of names

getPlace_Organization ( )
    gets the place and organization related to the post
    Returns food organization, location

gsearch = <context.googlesearch.GoogleSearch object>

nlp = <spacy.lang.en.English object>

person = "

```

## 1.15 gameextract module

```

class gameextract.GameExtractor
    Bases: object
    Context Understanding for The Gamer

    activities ( post )
        extracts data for "narrating an activity" subject situation
        param post post
        Returns

    activitiesDone = []

    activitiesTerms = ['giveaway', 'giveaways', 'livestreaming', 'livestreams', 'livestream']

    activityExists ( af, g )
        searches if activity with the same value already exists
        param af activity value
        param g game value
        Returns True or False

    addToActivities ( activity, game, action )
        add values to activities
        param activity activity
        param game game
        param action action
        Returns

    addToGames ( gamee, typee )
        add values to played games

```

**param gamee** game  
    **param typee** type  
**Returns**

**addToTeams** ( *team, games, country* )  
    add values to supported teams  
    **param team** team  
    **param games** games  
    **param country** country  
**Returns**

**api** = <context.alchemyapi.AlchemyAPI object>  
**asstypeinfo** = <context.asstypes.AssTypeInfo object>  
**conceptnet** = <context.conceptnet.ConceptNet object>  
**dbpedia** = <context.dbpedia.DBPedia object>  
**eventTerms** = ['festival', 'fest', 'concert', 'cup', 'tournament']  
**extractStory** ( *story* )  
    extract data from the event  
    **param story** event  
**Returns**

**gameExtract** ( *posts, persona, person* )  
    Main function for The Gamer context understanding  
    **param posts** array of cleaned posts  
    **param persona** persona  
    **param person** user  
**Returns** array of extracted information

**game\_dic** = <prepros.dictionaries.game\_dic.GameDictionary object>  
**game\_team** = <prepros.dictionaries.game\_team\_dic.GameTeamsDictionary object>  
**games** = {}  
**generateAssertions** ( )  
    collect extracted information  
**Returns**

**gsearch** = <context.googlesearch.GoogleSearch object>  
**nlp** = <spacy.lang.en.English object>  
**peopleEntities** = ['gamer', 'streamer']  
**person** = "  
**persona** = "



**playOnPost** ( *post* )

extracts the game he plays from the shared post caption

**param post** shared post caption

**Returns**

**playOnSentiment** ( *sentiment* )

extracts the game he plays from the user thought

**param sentiment** user thought

**Returns**

**playOnStory** ( *storyDict* )

extracts the game he plays from the event

**param storyDict** event data

**Returns**

**subjects** = {}

**teamOnPost** ( *post* )

extracts the team he supports from the shared post caption

**param post** post

**Returns**

**teamOnSentiment** ( *sentiment* )

extracts the team he supports from the user thought

**param sentiment** user thought

**Returns**

**teamOnStory** ( *storyDict* )

extracts the team he supports from the event

**param storyDict** event data

**Returns**

**teams** = {}

**terms** = ['gaming', 'game']

## 1.16 googlesearch module

**class** googlesearch.**GoogleSearch**

Bases: `object`

**dataCleaning** ( *description* )

a part of the process of getting the frequent words

**param description**

description

**Returns**

**feature\_extraction** ( *train\_texts* )

a part of the process of getting the frequent words

**param train\_texts** train\_texts

**Returns**

**getFood** ( *mentions* )

searches through Google and determines if it is a food

**param mentions** mentions

**Returns**

**get\_most\_common\_terms** ( *train\_texts* )

a part of the process of getting the frequent words

**param train\_texts** train\_texts

**Returns**

**isPerson** ( *query* )

searches through Google and determines if it is a person

**param query** query

**Returns** True or False

**search** ( *query* )

searches through Google

**param query** query

**Returns** 30 most frequent words

**stemming\_tokenizer** ( *text* )

a part of the process of getting the frequent words

**param text** text

**Returns**

**tf\_idf** ( *description* )

a part of the process of getting the frequent words

**param description**

description

**Returns**

## 1.17 likedpageevents module

**class** likedpageevents.LikedPagesEvents

Bases: object

**extractEvents** ( *data, persona* )

extracts events

**param data** events

**param persona** persona

**Returns** array of events in data frames

**extractLikedPages** ( *data, persona* )

extracts liked page

**param data**     liked pages

**param persona** persona

Returns array of liked pages in data frames

**fanCateg** = ['Automotive Company', 'Biotechnology Company', 'Cargo & Freight Company', 'Community Organization', 'Community Services', 'Company', 'Health/Beauty', 'Non-Governmental Organization', 'Non-Profit Organization', 'Organization', 'Political Organization', 'Political Party', 'Retail Company', 'Telecommunication Company', 'Tobacco Company', 'Travel Company', 'App Page', 'Appliances', 'Baby Goods/Kids Goods', 'Bags/Luggage', 'Brand', 'Building Materials', 'Camera/Photo', 'Cars', 'Clothing (Brand)', 'Commercial Equipment', 'Furniture', 'Home Décor', 'Household Supplies', 'Jewelry/Watches', 'Kitchen/Cooking', 'Office Supplies', 'Patio/Garden', 'Pet Supplies', 'Pharmaceuticals', 'Phone/Tablet', 'Product/Service', 'Software', 'Tools/Equipment', 'Vitamins/Supplements', 'Website', 'Wine/Spirits', 'Actor', 'Artist', 'Author', 'Band', 'Blogger', 'Chef', 'Comedian', 'Dancer', 'Entrepreneur', 'Fashion Model', 'Fictional Character', 'Film Director', 'Fitness Model', 'Government Official', 'Journalist', 'Motivational Speaker', 'Movie Character', 'Musician', 'News Personality', 'Pet', 'Photographer', 'Political Candidate', 'Politician', 'Producer', 'Public Figure', 'Scientist', 'Teacher', 'Video Creator', 'Writer', 'Album', 'Book', 'Book Series', 'Book Store', 'Concert Tour', 'Festival', 'Fictional Character', 'Library', 'Literary Arts', 'Magazine', 'Movie', 'Movie Character', 'Movie Theater', 'Movie/Television Studio', 'Music Award', 'Music Chart', 'Music Video', 'Performance & Event Venue', 'Performance Art', 'Performing Arts', 'Podcast', 'Radio Station', 'Record Label', 'Show', 'Song', 'Theatrical Play', 'Theatrical Productions', 'TV Channel', 'TV Network', 'TV Show', 'TV/Movie Award']

**foodieCateg** = ['Agriculture Company', 'Company', 'Food & Beverage Company', 'Retail Company', 'Tobacco Company', 'Non-Governmental Organization', 'Non-Profit Organization', 'Organization', 'Retail Company', 'App Page', 'Brand', 'Food & Beverage Company', 'Chef']

**gamerCateg** = ['Community Organization', 'Community Services', 'Company', 'Computer Company', 'Internet Company', 'Non-Governmental Organization', 'Non-Profit Organization', 'Organization', 'Retail Company', 'App Page', 'Board Game', 'Brand', 'Computers (Brand)', 'Electronics', 'Games/Toys', 'Phone/Tablet', 'Product/Service', 'Software', 'Video Game', 'Amateur Sports Team', 'Sports League', 'Sports Team', 'Stadium', 'Arena & Sports Venue', 'School Sports Team', 'Athlete', 'Coach']

**getEventDetails** ( *page* )

gets the event details such as event name and rsvp from the event

**param page** page

Returns event, rsvp

**getPageName** ( *page* )

gets the page name from a liked page

**param page** page

Returns page name

**nlp** = <spacy.lang.en.English object>

**sportsCateg** = ['Community Organization', 'Community Services', 'Company', 'Non-Governmental Organization', 'Non-Profit Organization', 'Organization', 'Retail Company', 'App Page', 'Brand', 'Amateur Sports Team', 'School Sports Team', 'Sports League', 'Sports Team', 'Stadium', 'Arena & Sports Venue', 'Athlete', 'Coach']

## 1.18 sportextract module

**class** sportextract.**SportExtractor**

Bases: `object`

Context Understanding for The Sports Fanatic

**addToFanOf** ( *fanOf*, *categories* )

add values to fanOf

**param fanOf** fanOf

**param categories** categories

Returns

**addToTeam** ( *team*, *categories* )

add values to teams

**param team** team

**param categories** categories

Returns

**combinePost** ( *p* )

combines the story, post, and sentiment

**param p** post instance

Returns

**determineSport** ( *post* )

distinguishes if it is a sport or not

**param post** combined post

Returns

**extractFanOf** ( *statement* )

extracts the athlete he supports

**param statement** user thoughts

Returns

**extractStory** ( *story* )

extracts data from the event

**param story** event

Returns event data

**generateAssertions** ( )

collect extracted information

Returns

**getSentiment** ( *statement* )

gets Sentiment from Watson API

**param statement** user thought

Returns

**getSubjects** ( *statement* )

determines the subject

**param statement** user thought

**Returns**

**searchLeague** ( *page* )

extracts the leagues he follows

**param page** shared post caption

**Returns**

**sportExtract** ( *posts, persona, person* )

Main function for The Sports Fanatic context understanding

**param posts** array of cleaned posts

**param persona** persona

**param person** user

**Returns** array of extracted information

## 1.19 translator module

**class** translator.**TGENTranslator**

Bases: `object`

**translateQuery** ( *query* )

translates Tagalog to English texts

**param query** query

**Returns** translated text

- [Index](#)
- [Module Index](#)
- [Search Page](#)



## a

alchemyapi, 5  
asstypes, 6

## c

clean, 2  
conceptnet, 7  
contextunderstanding, 7

## d

database, 1  
dbpedia, 8  
driver, 1

## e

emojiremove, 3  
entity, 4

## f

fanextract, 8  
filemanager, 2  
foodextract, 10

## g

gameextract, 11  
googlesearch, 13

## l

likedpagesevents, 14

## s

spellcheck, 4  
sportextract, 16

## t

translator, 17





## A

activities() (gameextract.GameExtractor method), 11  
activitiesDone (gameextract.GameExtractor attribute), 11  
activitiesTerms (gameextract.GameExtractor attribute), 11  
activityExists() (gameextract.GameExtractor method), 11  
addToActivities() (gameextract.GameExtractor method), 11  
addToFanOf() (sportextract.SportExtractor method), 16  
addToGames() (gameextract.GameExtractor method), 11  
addToTeam() (sportextract.SportExtractor method), 16  
addToTeams() (gameextract.GameExtractor method), 12  
AlchemyAPI (class in alchemyapi), 5  
alchemyapi (module), 5  
api (foodextract.FoodieExtractor attribute), 10  
api (gameextract.GameExtractor attribute), 12  
AssTypeInfo (class in asstypes), 6  
asstypeinfo (contextunderstanding.ContextUnderstanding attribute), 7  
asstypeinfo (foodextract.FoodieExtractor attribute), 10  
asstypeinfo (gameextract.GameExtractor attribute), 12  
asstypes (module), 6

## C

Clean (class in clean), 2  
clean (module), 2  
cleanData() (clean.Clean method), 2  
combinePost() (sportextract.SportExtractor method), 16  
compareWord() (spellcheck.SpellChecker method), 4  
ConceptNet (class in conceptnet), 7

conceptnet (foodextract.FoodieExtractor attribute), 10  
conceptnet (gameextract.GameExtractor attribute), 12  
conceptnet (module), 7  
connectDB() (database.Database method), 1  
ContextUnderstanding (class in contextunderstanding), 7  
contextunderstanding (module), 7  
createAssertionStoryPost() (fanextract.FanExtractor method), 8

## D

data (alchemyapi.AlchemyAPI attribute), 5  
Database (class in database), 1  
database (module), 1  
dataCleaning() (googlesearch.GoogleSearch method), 13  
DBPedia (class in dbpedia), 8  
dbpedia (foodextract.FoodieExtractor attribute), 10  
dbpedia (gameextract.GameExtractor attribute), 12  
dbpedia (module), 8  
describeSubject() (fanextract.FanExtractor method), 9  
determineSport() (sportextract.SportExtractor method), 16  
dividePost() (clean.Clean method), 2  
doesExists() (entity.Entity method), 4  
Driver (class in driver), 1  
driver (module), 1

## E

EmojiRemove (class in emojiremove), 3  
emojiremove (module), 3  
Entity (class in entity), 4  
entity (module), 4  
eventTerms (gameextract.GameExtractor attribute), 12  
extractElements() (clean.Clean method), 3  
extractEvents() (likedpagesevents.LikedPage-

sEvents method), 14  
 extractFanOf() (sportextract.SportExtractor method), 16  
 extractFoodPostInformation() (foodextract.FoodieExtractor method), 10  
 extractLikedPages() (likedpageevents.LikedPagesEvents method), 15  
 extractPost() (fanextract.FanExtractor method), 9  
 extractSentiment() (fanextract.FanExtractor method), 9  
 extractStory() (fanextract.FanExtractor method), 9  
 extractStory() (gameextract.GameExtractor method), 12  
 extractStory() (sportextract.SportExtractor method), 16  
 extractStoryInformation() (foodextract.FoodieExtractor method), 10

## F

fan (contextunderstanding.ContextUnderstanding attribute), 7  
 fanCateg (likedpageevents.LikedPagesEvents attribute), 15  
 fanextract (module), 8  
 fanExtract() (fanextract.FanExtractor method), 9  
 FanExtractor (class in fanextract), 8  
 fangirlboy() (asstype.AssTypeInfo method), 6  
 feature\_extraction() (googlesearch.GoogleSearch method), 14  
 FileManager (class in filemanager), 2  
 filemanager (module), 2  
 find\_adj() (fanextract.FanExtractor method), 9  
 find\_root() (foodextract.FoodieExtractor method), 10  
 find\_verb() (fanextract.FanExtractor method), 9  
 findMostSimilar() (spellcheck.SpellChecker method), 4  
 findSubject() (fanextract.FanExtractor method), 9  
 findTitleInQuotes() (fanextract.FanExtractor method), 9  
 foodextract (module), 10  
 foodie (contextunderstanding.ContextUnderstanding attribute), 7  
 foodie() (asstype.AssTypeInfo method), 6  
 foodieCateg (likedpageevents.LikedPagesEvents attribute), 15  
 foodieExtract() (foodextract.FoodieExtractor method), 10  
 FoodieExtractor (class in foodextract), 10

## G

game (contextunderstanding.ContextUnderstanding attribute), 7  
 game\_dic (gameextract.GameExtractor

attribute), 12  
 game\_team (gameextract.GameExtractor attribute), 12  
 gameextract (module), 11  
 gameExtract() (gameextract.GameExtractor method), 12  
 GameExtractor (class in gameextract), 11  
 gamerCateg (likedpageevents.LikedPagesEvents attribute), 15  
 games (gameextract.GameExtractor attribute), 12  
 generateAssertions() (gameextract.GameExtractor method), 12  
 generateAssertions() (sportextract.SportExtractor method), 16  
 get\_most\_common\_terms() (googlesearch.GoogleSearch method), 14  
 getAssertion() (asstype.AssTypeInfo method), 7  
 getCategories() (alchemyapi.AlchemyAPI method), 5  
 getCategoriesNoRelevance() (alchemyapi.AlchemyAPI method), 5  
 getCategoriesNoRelevance2() (alchemyapi.AlchemyAPI method), 5  
 getChecklist() (asstype.AssTypeInfo method), 7  
 getConcepts() (alchemyapi.AlchemyAPI method), 5  
 getConceptsRelevance() (alchemyapi.AlchemyAPI method), 5  
 getContext() (contextunderstanding.ContextUnderstanding method), 7  
 getDateTime() (database.Database method), 1  
 getEntities() (alchemyapi.AlchemyAPI method), 5  
 getEventDetails() (likedpageevents.LikedPagesEvents method), 15  
 getEventsByLabel() (database.Database method), 1  
 getFood() (foodextract.FoodieExtractor method), 10  
 getFood() (googlesearch.GoogleSearch method), 14  
 getFoodType() (foodextract.FoodieExtractor method), 11  
 getHypernym() (dbpedia.DBpedia method), 8  
 getIsTypeOf() (conceptnet.ConceptNet method), 7  
 getKeywords() (alchemyapi.AlchemyAPI method), 6  
 getKeywordsNoRelevance() (alchemyapi.AlchemyAPI method), 6  
 getLikesByLabel() (database.Database method), 2  
 getPageName() (likedpageevents.LikedPagesEvents method), 15  
 getPersonMentioned() (foodextract.FoodieEx-

tractor method), 11  
 getPlace\_Organization() (foodextract.FoodieExtractor method), 11  
 getPostsByLabel() (database.Database method), 2  
 getProfile() (database.Database method), 2  
 getRelations() (alchemyapi.AlchemyAPI method), 6  
 getSemanticRoles() (alchemyapi.AlchemyAPI method), 6  
 getSentiment() (alchemyapi.AlchemyAPI method), 6  
 getSentiment() (fanextract.FanExtractor method), 9  
 getSentiment() (sportextract.SportExtractor method), 16  
 getSubjects() (sportextract.SportExtractor method), 17  
 getType() (dbpedia.DBpedia method), 8  
 GoogleSearch (class in googlesearch), 13  
 googlesearch (module), 13  
 gsearch (foodextract.FoodieExtractor attribute), 11  
 gsearch (gameextract.GameExtractor attribute), 12

## I

init() (driver.Driver method), 1  
 inPeople() (spellcheck.SpellChecker method), 4  
 isAnEntity() (entity.Entity method), 4  
 isArt() (fanextract.FanExtractor method), 9  
 isPerson() (googlesearch.GoogleSearch method), 14

## L

LikedPagesEvents (class in likedpagesevents), 14  
 likedpagesevents (module), 14

## M

mild\_clean() (clean.Clean method), 3

## N

narratingActivity() (fanextract.FanExtractor method), 10  
 natural\_language\_understanding (alchemyapi.AlchemyAPI attribute), 6  
 nlp (dbpedia.DBpedia attribute), 8  
 nlp (foodextract.FoodieExtractor attribute), 11  
 nlp (gameextract.GameExtractor attribute), 12  
 nlp (likedpagesevents.LikedPagesEvents attribute), 15  
 nlu() (alchemyapi.AlchemyAPI method), 6

## P

peopleEntities (gameextract.GameExtractor

attribute), 12  
 person (foodextract.FoodieExtractor attribute), 11  
 person (gameextract.GameExtractor attribute), 12  
 persona (gameextract.GameExtractor attribute), 12  
 playOnPost() (gameextract.GameExtractor method), 13  
 playOnSentiment() (gameextract.GameExtractor method), 13  
 playOnStory() (gameextract.GameExtractor method), 13

## R

remove\_emoji() (emojiremove.EmojiRemove method), 3  
 remove\_specialChars() (emojiremove.EmojiRemove method), 3  
 removeExtras() (emojiremove.EmojiRemove method), 3  
 removeNewLine() (emojiremove.EmojiRemove method), 3  
 removeWhitespaces() (emojiremove.EmojiRemove method), 3

## S

search() (conceptnet.ConceptNet method), 7  
 search() (dbpedia.DBpedia method), 8  
 search() (googlesearch.GoogleSearch method), 14  
 searchFoodCategory() (dbpedia.DBpedia method), 8  
 searchLeague() (sportextract.SportExtractor method), 17  
 sparql (dbpedia.DBpedia attribute), 8  
 spell() (spellcheck.SpellChecker method), 4  
 spell\_Eng() (spellcheck.SpellChecker method), 5  
 spell\_Fil() (spellcheck.SpellChecker method), 5  
 spellcheck (module), 4  
 SpellChecker (class in spellcheck), 4  
 spellCorrect() (spellcheck.SpellChecker method), 5  
 sport (contextunderstanding.ContextUnderstanding attribute), 7  
 sportextract (module), 16  
 sportExtract() (sportextract.SportExtractor method), 17  
 SportExtractor (class in sportextract), 16  
 sportsCateg (likedpagesevents.LikedPagesEvents attribute), 15  
 stemming\_tokenizer() (googlesearch.GoogleSearch method), 14  
 subjects (gameextract.GameExtractor attribute), 13

## T

`teamOnPost()` (`gameextract.GameExtractor` method), [13](#)  
`teamOnSentiment()` (`gameextract.GameExtractor` method), [13](#)  
`teamOnStory()` (`gameextract.GameExtractor` method), [13](#)  
`teams` (`gameextract.GameExtractor` attribute), [13](#)  
`terms` (`gameextract.GameExtractor` attribute), [13](#)  
`tf_idf()` (`googlesearch.GoogleSearch` method), [14](#)  
`TGENTranslator` (class in `translator`), [17](#)  
`thorough_clean()` (`clean.Clean` method), [3](#)  
`translateQuery()` (`translator.TGENTranslator` method), [17](#)  
`translator` (module), [17](#)

## W

`writeFile()` (`filemanager.FileManager` method), [2](#)