Name- ASHWIN CHAUDHARI
Internship Program-ARTIFICIAL INTELLIGENCE
Batch- APRIL 2022- AUGUST 2022
Certificate Code-TCRIL01R31
Date of submission-20-08-2022

Technical Coding Research Innovation, Navi
Mumbai, Maharashtra, India-410206

ENTITY DETERMINATION IN THE FOOD DELIVEY DATA USING SPACY'S NER

A Case-Study Submitted for the requirement of **Technical Coding Research Innovation**

For the Internship Project work done during
ARTIFICIAL INTELLIGENCE INTERNSHIP PROGRAM

by
ASHWIN CHAUDHARI (TCRIL01R31)

Rutuja Doiphode
CO-FOUNDER &CEO
TCR innovation.

Name- ASHWIN CHAUDHARI
Internship Program-ARTIFICIAL INTELLIGENCE
Batch- APRIL 2022- AUGUST 2022
Certificate Code-TCRIL01R31
Date of submission-20-08-2022

Abstract –

Food Computing is currently a fast-growing field of research. Natural language processing (NLP) is also increasingly essential in this field, especially for recognising food entities.
However, there are still only a few well-defined tasks that serve as benchmarks for solutions in this area. We introduce a new dataset – called FoodData central – to bridge this gap. In this dataset, Named Entity Recognition (NER) models are expected to find or infer various types of entities helpful in processing recipes, e.g. food products, quantities and their units, names of cooking processes, physical quality of ingredients, their purpose, taste.
The dataset consists of more than 30,000 entities to extract. It is comprised of a small number of rules based on computational linguistics and semantic information that describe the food entities. Experimental results from the evaluation performed using two different datasets showed that very promising results can be achieved. The proposed method achieved 97% precision, 94% recall, and 96% F1 score. We share the dataset and the task to encourage progress on more in-depth and complex information extraction from recipes.

Index Terms -

I. Introduction

In the field of natural language processing there is the Information Extraction task, which consists in elicitation of data relevant to a particular topic from non-structured texts. One of the subtasks of Information Extraction is the recognition and extraction of named entities (Named Entity Recognition, NER). The results of recognition and classification of proper nouns in a text document are widely used in information retrieval, machine translation, question answering and automatic summarization.In this paper, we focus on IE of food entities. To the best of our knowledge, not a large amount of re- search focusing on food entities has been done. How- ever, nowadays, the knowledge about extracted food entities and their relations with other biomedical enti- ties (like genes, drugs, diseases, etc.) is important for improving public health.

The main contributions of this paper are:

• A rule-based NER method for IE of food entities.
• Evaluation of the proposed method, which pro- vides promising results on unstructured data, without a need for an annotated corpus.

About dataset :

We'll be using food data from the USDA's Branded Food's dataset.we will import two csv files:
1. food.csv
2.npr.csv

	fdc_id	data_type	description	food_category_id	publication_date
0	356425	branded_food	MOCHI ICE CREAM BONBONS	NaN	2019-04-01
1	356426	branded_food	CHIPOTLE BARBECUE SAUCE	NaN	2019-04-01
2	356427	branded_food	HOT & SPICY BARBECUE SAUCE	NaN	2019-04-01
3	356428	branded_food	BARBECUE SAUCE	NaN	2019-04-01
4	356429	branded_food	BARBECUE SAUCE	NaN	2019-04-01

	Article
0	In the Washington of 2016, even when the polic...
1	Donald Trump has used Twitter — his prefe...
2	Donald Trump is unabashedly praising Russian...
3	Updated at 2:50 p. m. ET, Russian President Vl...
4	From photography, illustration and video, to d...

II. Case Study

Title: Entity Determination In The Food Delivery Data Using spaCy's NER

Tools used: Jupyter Notebook, python

III. Theory:

Python: Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

Pandas: Pandas is an open-source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named NumPy, which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other data science modules inside the Python ecosystem, and is typically included in every Python distribution.

Matplotlib: Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals.Matplotlib consists of several plots like line, bar, scatter, histogram etc.

spaCy : spaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython. spaCy is designed specifically for production use and helps you build applications that process and “understand” large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning.SpaCy can be installed using a simple pip install. Using pip, spaCy releases are available as source packages and binary wheels. Before you install spaCy and its dependencies, make sure that your pip, setuptools and wheel are up to date.

NER : Named-entity recognition (NER) is the method or system of extracting information which allows us to properly understand the subject or topic of the raw text. It is a process of automatically identifying the named entities present in a given text of any documents and classifying them into predefined categories such as ‘person’, ‘organization’, ‘location’ and so on.

The goal of a named entity recognition (NER) system is to identify all textual mentions of the named entities and also classify them. It helps to solve many real world problems in Natural Language Processing (NLP). Commonly Used Types of Named Entity are :

ORGANIZATION: Georgia-Pacific Corp., WHO

PERSON: Eddy Bonte, President Obama

LOCATION: Murray River, Mount Everest

DATE: June, 2008–06–29

TIME: two fifty a m, 1:30 p.m.

MONEY: 175 million Canadian Dollars, GBP 10.40

PERCENT: twenty pct, 18.75 %

FACILITY: Washington Monument, Stonehenge

GPE: South East Asia, Midlothian

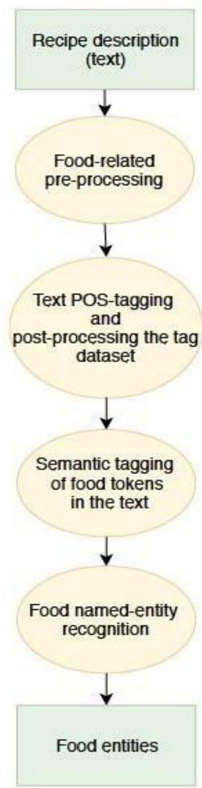
Name- ASHWIN CHAUDHARI
Internship Program-ARTIFICIAL INTELLIGENCE
Batch- APRIL 2022- AUGUST 2022
Certificate Code-TCRIL01R31
Date of submission-20-08-2022

Food-related Text Pre-processing:

To enable food-named entity recognition, in this pa- per, we propose a rule-based approach, called FoodIE. It works with unstructured data (more specifically, with a recipe that includes textual data in form of in- structions on how to prepare the dish) and consists of four steps:

• Food-related text pre-processing
• Text POS-tagging and post-processing of the tag dataset .Semantic tagging of food tokens in the text
• Food-named entity recognition

The flowchart of the methodology is presented in Further, we are going to explain each part in more detail.



Food-named Entity Recognition

To obtain food chunks, we used the modified tag set from the USAS semantic tagger obtained in Subsec- tion 3.2 in combination with the food tokens obtained in Subsection 3.3. The process of food-named entity recognition consists of three steps.

Then, to determine if it truly is a food entity chunk or just a chunk related to food but not a food entity in and of itself, we check the last token of the chunk. The whole chunk is discarded if the last token is:

• A noun (starts with NN) and a general non-food object, or
• in the disallowed category as defined by the rule engine, or
• in the disallowed category as defined by the resources

Text POS-tagging and Post-processing :

To obtain the morphological information from a tex- tual data, we use UCREL Semantic Analysis System (USAS) and coreNLP.
The USAS semantic tagger provides word tokens associated with their POS tags, lemmas, and seman- tic tags. The semantic tags show semantic fields that group together word senses that are related at some level of generality with the same contextual concept. The groups include not only synonyms and antonyms Furthermore, the same is done using the coreNLP library, which includes all of the above except seman- tic tags.

IV. Evaluation Matrics

There are different approaches for the evaluation of NER systems [1]. Performance evaluation characterizes the ability of a tool to find the boundaries of a named entity and correctly determine its type. The score can be computed for exact and partial matching. The exact matching assumes the exact coincidence of the boundaries of the predicted and true named entity. Under such conditions the systems of the participants in the competition held as part of the CoNLL-2003 conference [2] were evaluated. However, in some cases the exact matching of boundaries is not as important as the identification of a major part of a named entity. For example, the phrases “The United States” and “United States” are almost similar and differ only in the presence of the article. At the MUC (Message Understanding Conference) [19], metrics were used to evaluate the systems, taking into account the partial overlap of the predicted andtrue named entities.

In this paper systems are evaluated under two conditions:
1) Exact matching of boundaries and types of predicted and true entities.
2) Partial matching of the predicted and true entities when the types coincide and the boundaries of the predicted entity are inside the boundaries of the true one, or vice versa.

To evaluate the performance of NER the following metrics are used, calculated with respect to type i:

- precision – the proportion of correctly classified entities (True Positives) among all entities assigned by the classifier to type i (True Positives and False Positives)

$$P_i = \frac{TP_i}{TP_i + FP_i}; \qquad (1)$$

- recall – the proportion of correctly classified entities (True Positives) among all entities belonging to type i (True Positives and False Negatives)

$$R_i = \frac{TP_i}{TP_i + FN_i}; \qquad (2)$$

- F1-score – harmonic mean of precision and recall

V . Analysis and Results :

Preparing the food data

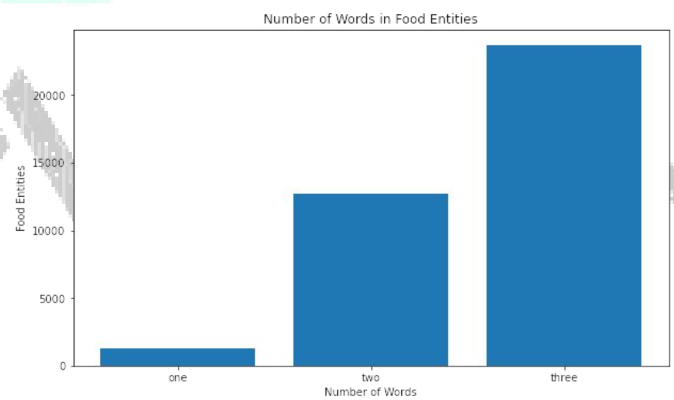
```
# read in the food csv file
food_df = pd.read_csv("food.csv")

# print row and column information
food_df.head()
```

	fdc_id	data_type	description	food_category_id	publication_date
0	356425	branded_food	MOCHI ICE CREAM BONBONS	NaN	2019-04-01
1	356426	branded_food	CHIPOTLE BARBECUE SAUCE	NaN	2019-04-01
2	356427	branded_food	HOT & SPICY BARBECUE SAUCE	NaN	2019-04-01
3	356428	branded_food	BARBECUE SAUCE	NaN	2019-04-01
4	356429	branded_food	BARBECUE SAUCE	NaN	2019-04-01

Now, we need to think about how we want our data to be distributed. By reducing to 3-worded food items, we effectively have food entities that look like this:

hamburger | 1-worded
grilled cheese | 2-worded
chocolate ice cream | 3-worded



Name- ASHWIN CHAUDHARI
Internship Program-ARTIFICIAL INTELLIGENCE
Batch- APRIL 2022- AUGUST 2022
Certificate Code-TCRIL01R31
Date of submission-20-08-2022

Split train and test food data : We'll break up our food sentences (which contain our entities) into a training set and a test set. We also need the data to be in a specific format for training:

```
# create dictionaries to store the generated food combinations. Do note that one_food != one_worded_food. one_food == "barbecue s
TRAIN_FOOD_DATA = {
    "one_food": [],
    "two_foods": [],
    "three_foods": []
}

TEST_FOOD_DATA = {
    "one_food": [],
    "two_foods": [],
    "three_foods": []
}

# one_food, two_food, and three_food combinations will be limited to 167 sentences
FOOD_SENTENCE_LIMIT = 167

# helper function for deciding what dictionary and subsequent array to append the food sentence on to
def get_food_data(count):
    return {
        1: TRAIN_FOOD_DATA["one_food"] if len(TRAIN_FOOD_DATA["one_food"]) < FOOD_SENTENCE_LIMIT else TEST_FOOD_DATA["one_food"],
        2: TRAIN_FOOD_DATA["two_foods"] if len(TRAIN_FOOD_DATA["two_foods"]) < FOOD_SENTENCE_LIMIT else TEST_FOOD_DATA["two_foods
        3: TRAIN_FOOD_DATA["three_foods"] if len(TRAIN_FOOD_DATA["three_foods"]) < FOOD_SENTENCE_LIMIT else TEST_FOOD_DATA["three
    }[count]

# the pattern to replace from the template sentences
pattern_to_replace = "{}"

# shuffle the data before starting
foods = foods.sample(frac=1)

# the count that helps us decide when to break from the for loop
food_entity_count = foods.size - 1

# start the while loop, ensure we don't get an index out of bounds error
while food_entity_count >= 2:
    entities = []

    # pick a random food template
    sentence = food_templates[random.randint(0, len(food_templates) - 1)]

    # find out how many braces "{}" need to be replaced in the template
    matches = re.findall(pattern_to_replace, sentence)

    # for each brace, replace with a food entity from the shuffled food data
    for match in matches:
        food = foods.iloc[food_entity_count]
        food_entity_count -= 1
```

Generating Revision Data :

As mentioned in the overview, we also need to generate sentences that contain spaCy entities. This helps us avoid the situation where the NER model is able to identify the FOOD entities, but forgets how to classify entities like ORG or PERSON.

While ORG or PERSON isn't important for nutrition-tracking, other entities like QUANTITY and CARDINAL will help us associate foods with their quantities later on:
I ate two slices of toast.

Preparing the revision data

```
# read in the revision data (just used a random article dataset
npr_df = pd.read_csv("npr.csv")
ER
# print row and column information
npr_df.head()
```

	Article
0	In the Washington of 2016, even when the polic...
1	Donald Trump has used Twitter — his prefe...
2	Donald Trump is unabashedly praising Russian...
3	Updated at 2:50 p. m. ET, Russian President Vl...
4	From photography, illustration and video, to d...

Split train and test revision data : When splitting the train and test data, we'll ensure that the revision training data has at least 100 examples of the different entity types.

```
# create arrays to store the revision data
TRAIN_REVISION_DATA = []
TEST_REVISION_DATA = []

# create dictionaries to keep count of the different entities
TRAIN_ENTITY_COUNTER = {}
TEST_ENTITY_COUNTER = {}

# This will help distribute the entities (i.e. we don't want 1000 PERSON entities, but only 80 ORG entities)
REVISION_SENTENCE_SOFT_LIMIT = 100

# helper function for incrementing the revision counters
def increment_revision_counters(entity_counter, entities):
    for entity in entities:
        label = entity[2]
        if label in entity_counter:
            entity_counter[label] += 1
        else:
            entity_counter[label] = 1

random.shuffle(revisions)
for revision in revisions:
    # get the entities from the revision sentence
    entities = revision[1]["entities"]

    # simple hack to make sure spaCy entities don't get too one-sided
    should_append_to_train_counter = 0
    for _, _, label in entities:
        if label in TRAIN_ENTITY_COUNTER and TRAIN_ENTITY_COUNTER[label] > REVISION_SENTENCE_SOFT_LIMIT:
            should_append_to_train_counter -= 1
        else:
            should_append_to_train_counter += 1

    # simple switch for deciding whether to append to train data or test data
    if should_append_to_train_counter >= 0:
        TRAIN_REVISION_DATA.append(revision)
        increment_revision_counters(TRAIN_ENTITY_COUNTER, entities)
    else:
        TEST_REVISION_DATA.append(revision)
        increment_revision_counters(TEST_ENTITY_COUNTER, entities)
```

Training the NER Model : For every food sentence, I have revision sentences. I haven't actually seen guidance on what this should be, so this is one of those "stir until good enough" moments

```
# add NER to the pipeline and the new label
ner = nlp.get_pipe("ner")
ner.add_label("FOOD")

# get the names of the components we want to disable during training
pipe_exceptions = ["ner", "trf_wordpiecer", "trf_tok2vec"]
other_pipes = [pipe for pipe in nlp.pipe_names if pipe not in pipe_exceptions]

# start the training loop, only training NER
epochs = 30
optimizer = nlp.resume_training()
with nlp.disable_pipes(*other_pipes), warnings.catch_warnings():
    warnings.filterwarnings("once", category=UserWarning, module='spacy')
    sizes = compounding(1.0, 4.0, 1.001)

    # batch up the examples using spaCy's minibatc
    for epoch in range(epochs):
        examples = TRAIN_DATA
        random.shuffle(examples)
        batches = minibatch(examples, size=sizes)
        losses = {}

        for batch in batches:
            texts, annotations = zip(*batch)
            nlp.update(texts, annotations, sgd=optimizer, drop=0.35, losses=losses)

        print("Losses ({}/{})".format(epoch + 1, epochs), losses)
```

Name- ASHWIN CHAUDHARI
Internship Program-ARTIFICIAL INTELLIGENCE
Batch- APRIL 2022- AUGUST 2022
Certificate Code-TCRIL01R31
Date of submission-20-08-2022

Evaluating the Model :

Evaluating the Model

```
]: # display sentence involving original entities
spacy.displacy.render(nlp("Apple is looking at buying U.K. startup for $1 billion"), style="ent")
```

Apple ORG is looking at buying U.K. GPE startup for $1 billion MONEY

```
]: # display sentences involving target entity
spacy.displacy.render(nlp("I had a hamburger and chips for lunch today."), style="ent")
spacy.displacy.render(nlp("I decided to have chocolate ice cream as a little treat for myself."), style="ent")
spacy.displacy.render(nlp("I ordered basmati rice, leaf spinach and cheese from Tesco yesterday"), style="ent")
```

I had a hamburger FOOD and chips FOOD for lunch today.

I decided to have chocolate ice cream FOOD as a little treat for myself.

I ordered basmati rice FOOD , leaf spinach FOOD and cheese FOOD from Tesco ORG yesterday

Initial results seem pretty good, let's evaluate on a wider scale.

Evaluating Food Entities: These results are really positive. We're stumbling with 1_worded_foods accuracy, though that's potentially because we had more testing data for 1_worded_foods. Perhaps with more test examples for 2_worded_foods and three_worded_foods, we'd also see that accuracy trend to ~91%.

```
for key in word_evaluation:
    correct = word_evaluation[key]["correct"]
    total = word_evaluation[key]["total"]

    print(f"{key}: {correct / total * 100:.2f}%")

food_total_sum = 0
food_correct_sum = 0

print("---")
for key in food_evaluation:
    correct = food_evaluation[key]["correct"]
    total = food_evaluation[key]["total"]

    food_total_sum += total
    food_correct_sum += correct

    print(f"{key}: {correct / total * 100:.2f}%")

print(f"\nTotal: {food_correct_sum/food_total_sum * 100:.2f}%")
```

```
1_worded_foods: 91.10%
2_worded_foods: 96.69%
3_worded_foods: 96.88%
---
one_food: 91.44%
two_foods: 94.76%
three_foods: 98.13%

Total: 94.14%
```

Evaluating Existing Entities:

These results are a little harder to interpret. After all, we're testing entities that the original spaCy model predicited for us. Those predicted entities may well be wrong since spaCy's accuracy is at around 86%. If 14% of the entities we're using to verify the accuracy of our new model are wrong, then where does that leave us.

A better comparison would be to load in spaCy's original model and use that to predict against this test set and compare that accuracy % to this one of 71%. We could then use that as a benchmark for measuring how introducing FOOD entities deteriorates our model.

```
sum_total = 0
sum_correct = 0

for entity in entity_evaluation:
    total = entity_evaluation[entity]["total"]
    correct = entity_evaluation[entity]["correct"]

    sum_total += total
    sum_correct += correct

    print("{} | {:.2f}%".format(entity, correct / total * 100))

print()
print("Overall accuracy: {:.2f}%".format(sum_correct / sum_total * 100))
```

```
PERSON | 80.18%
ORG | 50.84%
DATE | 68.61%
GPE | 82.56%
NORP | 83.61%
CARDINAL | 70.11%
QUANTITY | 79.53%
PERCENT | 88.44%
TIME | 50.88%
FAC | 56.58%
LOC | 68.69%
ORDINAL | 94.53%
MONEY | 84.11%
WORK_OF_ART | 58.78%
PRODUCT | 42.86%
EVENT | 63.46%
LANGUAGE | 91.67%
LAW | 75.00%

Overall accuracy: 71.23%
```

Name- ASHWIN CHAUDHARI
Internship Program-ARTIFICIAL INTELLIGENCE
Batch- APRIL 2022- AUGUST 2022
Certificate Code-TCRIL01R31
Date of submission-20-08-2022

Result : The results we arrived at is the following for our FOOD entities:

Category	Results
One-worded foods	91.10%
Two-worded foods	96.69%
Three-worded foods	96.88%
Sentences with one food	91.44%
Sentences with two foods	94.76%
Sentences with three foods	96.88%
Overall accuracy	**94.14%**

The results for our existing entities:

Category	Results
PERSON	80.18%
ORG	50.84%
DATE	68.61%
GPE	82.56%
NORP	83.61%
CARDINAL	70.11%
QUANTITY	79.53%
PERCENT	88.44%
TIME	50.88%
FAC	56.58%
LOC	68.69%
ORDINAL	94.53%
MONEY	84.11%
WORK_OF_ART	58.78%
PRODUCT	42.86%
EVENT	63.46%
LANGUAGE	91.67%
LAW	75.00%
Overall accuracy	**71.32%**

CONCLUSION:

So, In this project Branded Food's dataset. was analysed In this notebook, we have trained spaCy to identify FOOD entities from a body of text - a task known as named-entity recognition (NER). I decided to have chocolate ice cream as a little treat for myself. I had a hamburger and chips for lunch today. I ordered basmati rice, leaf spinach and cheese from Tesco yesterday.
spaCy has a NER accuracy of 85.85%, so something in that range would be nice for our FOOD entities. Overall accuracy came to be 71.23%

Name- ASHWIN CHAUDHARI
Internship Program-ARTIFICIAL INTELLIGENCE
Batch- APRIL 2022- AUGUST 2022
Certificate Code-TCRIL01R31
Date of submission-20-08-2022

REFERENCES:

Weiqing Min, Shuqiang Jiang, Linhu Liu, Yong Rui, and Ramesh Jain. A survey on food computing. ACM Comput. Surv., 52(5), sep 2019. ISSN 0360-0300. doi:10.1145/3329168. URL https://doi.org/10.1145/3329168.

Giulia Menichetti, Babak Ravandi, Dariush Mozaffarian, and Albert-László Barabási. Machine learning prediction of food processing. medRxiv, 2021. doi:10.1101/2021.05.22.21257615. URL https://www.medrxiv.org/content/ early/2021/05/25/2021.05.22.21257615.

Javier Marín, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Tor- ralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. IEEE Trans- actions on Pattern Analysis and Machine Intelligence, 43(1):187–203, 2021. doi:10.1109/TPAMI.2019.2927476.

Michal Bien, Michal Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In Brian Davis, Yvette Graham, John D. Kelleher, and Yaji Sripada, editors, Proceedings of the 13th International Conference on Natural Language Generation, INLG 2020, Dublin, Ireland, December 15-18, 2020, pages 22–28. Association for Computational Linguistics, 2020. URL https://aclanthology.org/2020.inlg-1.4/.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. Flow graph corpus from recipe texts. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14), pages 2370–2377, 2014.

Fernando De la Torre, Jessica Hodgins, Adam Bargteil, Xavier Martin, Justin Macey, Alex Collado, and Pep Beltran. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. robotics institute (2008), 2008.

Gorjan Popovski, Barbara Koroušic´ Seljak, and Tome Eftimov. FoodBase corpus: a new resource of annotated food entities. Database, 2019(baz121), January 2019a. ISSN 1758-0463. doi:10.1093/database/baz121. URL https://doi.org/10.1093/database/baz121.

Zara Nasar, Syed Waqar Jaffry, and Muhammad Kamran Malik. Named entity recognition and relation extraction: State-of-the-art. ACM Comput. Surv., 54(1):20:1–20:39, 2021. doi:10.1145/3445965. URL https://doi.org/10. 1145/3445965.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001, pages 282–289. Morgan Kaufmann, 2001.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. CoRR, abs/1508.01991, 2015. URL http://arxiv.org/abs/1508.01991.

Nadeesha Perera, Thi Thuy Linh Nguyen, Matthias Dehmer, and Frank Emmert-Streib. Comparison of text mining models for food and dietary constituent named-entity recognition. Machine Learning and Knowledge Extraction, 4 (1):254–275, 2022. ISSN 2504-4990. doi:10.3390/make4010012. URL https://www.mdpi.com/2504-4990/4/ 1/12.

Gjorgjina Cenikj, Gorjan Popovski, Riste Stojanov, Barbara Koroušic´ Seljak, and Tome Eftimov. BuTTER: Bidirec- Tional LSTM for Food Named-Entity Recognition. In 2020 IEEE International Conference on Big Data (Big Data), pages 3550–3556, December 2020. doi:10.1109/BigData50022.2020.9378151.

Gorjan Popovski, Stefan Kochev, Barbara Seljak, and Tome Eftimov. FoodIE: A Rule-based Named-entity Recognition Method for Food Information Extraction:. In Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods, pages 915–922, Prague, Czech Republic, 2019b. SCITEPRESS - Science and Technology Publications. ISBN 9789897583513. doi:10.5220/0007686309150922. URL http://www.scitepress.org/ DigitalLibrary/Link.aspx?doi=10.5220/0007686309150922.