Vrije Universiteit Amsterdam

Bachelor Thesis

Leveraging Multimodal Information Extraction for Automatic Knowledge Base Construction specific to Commonsense Knowledge in Cooking Recipes

---

Author: Leander Finck von Finckenstein (2640632)

1st supervisor:	supervisor name
daily supervisor:	supervisor name
2nd reader:	supervisor name

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

September 30, 2021

Abstract

This paper introduces 4 information extraction engines that extract commonsense knowledge from cooking recipes and store them via 7 edges in a knowledge base. The type of commonsense knowledge extracted is practical and 'how to' oriented, meaning that the utility of kitchen tools and their relation to food is investigated. For example, it is commonsense knowledge to use tongs to flip the rib-eye steak searing on the grill and then use a fork and a knife to eat it. Thus, the investigation aims to find out what actions a tool can perform, what tool is used to cook and eat food, and what pair of tools work well in combination with each other. To this end, a object detector is trained to detect 44 kitchen tools in recipe videos. In addition, SpaCy [1] and ConceptNet [7] are used to extract information from the texts. Both sources of data are produced by tasty.co. The average precision and recall of the created edges is 32.18% and 66.62% respectively. All code is available via github.com/leanderFinck1999/cooking_KB_construction.

1 Introduction

'Commonsense is the least common of all senses.' While it is debatable whether this is true for people, it certainly true for computer systems. If it were not, human-computer interaction would be greatly advanced and the livelong sci-fi dream of ever household owning a semi sentient robot that completes mundane chores would be a reality. The present has not yet caught up with the future because a system that exhibits general human-like commonsense knowledge is notoriously difficult to build.

General human-like commonsense means being able to reason and draw sound conclusions based on basic assumptions and previous experiences. Commonsense knowledge is often intuitive and almost never explicitly stated which makes it difficult to collect data for it. For example, if a recipe asks to 'fry the egg', its intuitive to use a pan. However, if the recipe were to say 'boil the egg', its obvious to use a pot. Paradoxically, it is clear which tool to use although it is never stated. Furthermore, the tool implied by the recipe also depends on the food mentioned although the action are the same. For example, its commonsense to peel a pineapple with a knife and peel a apple with a peeler. The examples come to show that the tools implied by the recipe depend on the action and the foods mentioned. Hence, commonsense knowledge specific to cooking recipes is defined by the ability to complete instructions efficiently with the most suitable tools, without them being explicitly stated.

The aim of this paper is therefore to make the implied knowledge found in cooking recipes explicit and store it in a knowledge base. Other knowledge bases representing commonsense knowledge, like ConceptNet, rely on crowed sourced and expert information as sources for their knowledge. Although the information obtained may be more accurate, it is time consuming and money intensive. Instead, this paper aims to build a knowledge base specific to cooking recipes automatically. The data used for this purpose originates from the tasty.co website, which offers thousands of baking and cooking recipes of various cuisines. Tasty.co also accompanies some of their recipes with short 1-3 minute videos which complement the text. To this end, computer vision is used to extract information from the videos, while natural language processing and other knowledge bases, like ConceptNet[7] and WordNet[6], are used to extract information from the text.

Nodes:	**Edges:**
• Foods: Ingredients, Dish, FoodConcept • Tools: Container, Utensil, Cutlery • Verbs & Antonyms	• *OperateWith* (Tool → Tool) • *EatWith* (Cutlery → Dish) • *Contains* (Container → Ingredients & FoodConcept) • *UsedToPrepare* (Utensil → Ingredients & FoodConcept) • *UsedFor* (Tools → Verbs & Antonyms) • *CookedBy* (FoodConcept → Verbs & Antonyms)

Table 1: Nodes represented in knowledge base and the corresponding edges

Since recipes are a sequence of instructions, the type of commonsense extracted is practical, meaning that kitchen tools are at the forefront of the investigation. The paper aims to find out which tools are used on which foods, which sets of tools are used in combination with each other and what actions the tool can perform. The kitchen tools are therefore the independent variable, so to speak, and the associated food, verb and antonym are the dependent variable. Antonyms are used from WordNet to define what a tool is not used for, which is useful since it gives the often neglected knowledge regarding negated information. Table 1 shows the list of nodes stored in the knowledge base and the semantic relations, i.e. edges, between them.

As discussed earlier, investigating kitchen tools is a challenge because the tools are implied by the text and rarely explicitly stated. It is human commonsense knowledge that decides which tool to use based on the verb and the food mentioned. The key to extracting such commonsense knowledge is therefore computer vision since tools are simply shown in the videos. For this to be meaningful, however, the parsing of the video and the text need to be synchronized so that the foods and verbs mentioned in the text can be mapped to a tool shown in the video. To overcome this challenge and others, simple heuristics were used.

First the data collection from the tasty.co website will be discussed in section 2, followed by a detailed explanation of the computer vision system build in section 3. Both sections lead to section 4 which is the main part of this paper. It discusses the information extraction process for each type of edge in the knowledge base and evaluates it based on its precision and recall. The paper then concludes the knowledge produced in the paper and evaluates it based on its reported precision and recall scores and wraps up the discussion with further research proposals.

2 Data Collection

As stated in the introduction, all data originates from the tasty.co website since it displays recipes in both video and text format. Scrapy was used to obtain the recipe in textual

format, while a combination of Selenium and Scrapy was used to load the video and scrape its content.

2.1 Scraping Recipes in Text Format

The website has a tasty.co/ingredient page where all recipes are categorized into their dominant ingredient and listed in alphabetical order. For example, the 'Chocolate' hyperlink in the tasty.co/ingredient page will lead to the tasty.co/ingredient/chocolate which contains 893 hyperlinks to recipes where chocolate is used, e.g. tasty.co/recipe/chocolate-chip-cookies is one of them. The web crawler will therefore start from the tasty.co/ingredient site and follow all hyperlinks that have 'topic', 'compilation', or 'recipe' in their URL. Once arrived at a recipe, the crawler parses the HTML and extract all text that correspond to the xpath of the recipe's preparation. This and other information, like the list of ingredients, found in the recipe are written to the recipe1.csv file. In this way, the web crawler managed to extract 2,065 recipes.

2.2 Scraping Recipes in Video Format

Inspection of a recipe page showed that the video displayed is loaded using AJAX, meaning that the videos cannot be scraped using Scrapy [3] alone since the python framework is not able to run JavaScript. To compensate for its shortcoming, Selenium is used which deploys a webdriver to open a dummy browser and run the JavaScript. Scrapy then parses the AJAX loaded content and scrapes the URL associated to the video. The URL is used by urllib.request.urlretrieve to store the video locally. In this way, a total of 449 videos are retrieved from the tasty.co website.

Note that out of the 2,065 recipes scraped, 1,828 come with a video although only 449 videos are retrieved, meaning that multiple recipes are mapped to the same video. This makes sense because loading the video gives a 206 HTTP response code, indicating that only part of the video is loaded. Unfortunately, each video needs to be associated with one text since one of the goals is to link the tools shown in the video to the verbs and foods mentioned in the text. This will not be possible with multiple texts mapping to the same video, since with the way it is currently stored, it is impossible to know which text corresponds to which part of the video. To make the mapping of text to video 1 to 1, recipes that link to the same video URL are filtered out, i.e. recipes remain if their video URL is unique. What is left are 340 recipes where one text is associated to one video.

2.3 Summary of the Data Collected

The list below summarizes the most important points made regarding the data retrieved:

- 2,065 recipes as text
- 449 recipes as video
- 340 recipes where 1 text maps to 1 video

3 Computer Vision

Computer Vision plays a vital role in this research since it shows the tools implied by the text, making it possible to obtain the commonsense knowledge automatically. Object detection is the most suitable type of computer vision for this task since the frame in the videos show multiple tools interacting with each other, all of which can to be identified and localized this way. Other types of computer vision like image classification are not an option since it only works if the image shows one tool to classify.

3.1 The TensorFlow Object Detection API and the COCO Data Set

TensorFlow offers a Object Detection API [2] with models pre-trained on the Common Objects in Context (COCO) [4] data set, which contains thousands of labelled images of 90 common every day objects in context (hence its name). Each of the objects fall under a category, among them are categories for food and kitchen objects. The included kitchen object are a fork, spoon, knife, bowl, bottle, wine glass and cup. Clearly, the list is not exhaustive. For this research to be meaningful, the object detector should be able to identify a number of basic tools used in the kitchen. To this end, a data set needs to be created which contains labelled images of kitchen tools used to train on.

3.2 Creating a Custom Object Detector

Fortunately, the TensorFlow Object Detection API makes it possible to create a custom object detector given a data set to train on. Just like the images in the COCO data set, the images in the custom data set should show the tools in context, i.e. show them in action. This is important because the object detector may not be able to detect a ladle full with soup if the images in the train set show the ladle being empty. Furthermore, tools in context has the advantage that the images show the tools from all possible angles which facilitates their identification and localization. The data set will therefore contain one second snapshot from some of the recipe videos previously retrieved off of the tasty.co site.

3.2.1 Preparing the Data Set

In total, the data set contains around 12,500 labelled images, 24,332 labels, for 44 kitchen tools. Table 2 shows the tools present in the data set with the corresponding number of labels.

The data set is split into a train and test set, where 90% of the images are used for training and the remaining 10% are used for testing. To perform the partition, the images in each class is split into the 9:1 ratio. This makes for better evaluation because the test set maintains the proportion of the classes contained in the data set. If the entire data set were partitioned into a 9:1 split, instead of each class, it could be that the test set does not contain some objects and a disproportionate amount of others. Even worse, it could be that all of the underrepresented tools are in the test set meaning there are no images of the tool in the train set for training.

This could be a real scenario because the data set as is already has great class imbalance. The object with the most labels is the bowl with 4,431 labels while the pizza-cutter has the

Count	Type of Tool	Tool	Train	Test	Total
1	Dishware	bowl	3988	443	4431
2	Container	pan	1616	179	1795
3	Container	pot	1402	155	1557
4	Utensil	pinch-bowl	984	109	1093
5	Container	cutting-board	774	85	859
6	Dishware	plate	738	81	819
7	Container	baking-sheet	731	81	812
8	Utensil	silicone-spatula	673	74	747
9	Utensil	jug	574	63	637
10	Utensil	lepel	514	57	571
11	Container	baking-dish	513	56	569
12	Utensil	tongs	490	54	544
13	Utensil	wooden-spatula	486	53	539
14	Utensil	whisk	461	51	512
15	Container	blender	451	50	501
16	Cutlery	fork	449	49	498
17	Cutlery	spoon	445	49	494
18	Cutlery	knife	436	48	484
19	Utensil	mixer	418	46	464
20	Dishware	food-container	396	43	439
21	Utensil	measuring-cup	392	43	435
22	Utensil	turner	351	39	390
23	Utensil	sieve	347	38	385
24	Container	baking-form	283	31	314
25	Utensil	peeler	282	31	313
26	Utensil	rolling-pin	267	29	296
27	Utensil	brush	260	28	288
28	Utensil	skimmer	253	28	281
29	Utensil	ladle	248	27	275
30	Utensil	scoop	217	24	241
31	Utensil	grater	215	23	238
32	Utensil	icing-spatula	213	23	236
33	Utensil	pepper-mill	201	22	223
34	Utensil	hammer	198	21	219
35	Cutlery	chopsticks	187	22	209
36	Utensil	oil-dispenser	184	20	204
37	Utensil	lid	178	19	197
38	Utensil	squeezer	172	19	191
39	Container	jar	170	18	188
40	Container	baking-rack	169	18	187
41	Utensil	oven-glove	162	15	177
42	Utensil	masher	160	17	177
43	Container	cupcake-tin	148	16	164
44	Utensil	pizza-cutter	126	13	139

Table 2: Table of the Object Detector's data set giving the detectable tools and the corresponding number of labels in both the train and test set.

least labels with just 139. The class imbalance occurs naturally, since almost every recipe uses a bowl to mix and combine ingredients while the pizza-cutter is only used to cut a pizza or bread at the end of a video. To even out the class imbalance, instead of using 1 frame per second in a video that show a pizza-cutter and other underrepresented objects, 20 frames per second are used which marginally increased the number of underrepresented objects in the data set.

3.2.2 Training

To train the object detector on the data set, transfer learning is applied on one of the pre-trained models offered by the TensorFlow API. The github.com/tensorflow/models page contains a list of models that use different object detection algorithm. The model used for this project is the SSD ResNet50 V1 FPN 640x640, which uses the state-of-the-art Single Shot MultiBox Detector algorithm (SSD)[5]. In contrast to other algorithms like the Faster-R-CNN algorithm, SSD's architecture allows for real-time object detection with an mean average precision (mAP) score of 76.9% on 300x300 sized images. For more details regarding SSD's architecture, see [citation]. The custom object detector is trained using a batch-size of 8 on a GPU made available by google colab, and training stopped when the total loss reached a value of about 0.4. The training progress of the individual loss functions can be seen in figure 5 in the appendix. A demo of the object detector can be seen in this video

3.2.3 Evaluation

Once the training process finished, the model is run on the test set and evaluated using the COCO evaluation metric. The metric consists of scores for the average recall/precision and the average recall/precision across scales.

The Average Precision score measures how accurate both the localization and the classification of the model is and is given by:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

where TP stands for True Positive and FP stands for False Positive. Predictions are classified as true positive if the overlap between the ground truth bounding box and the predicted bounding box is greater than a given threshold, otherwise it is a false positive. This overlap is a ratio called the Intersection over Union (IoU) and the evaluation metric uses thresholds of 0.5 and 0.75.

The Average Recall score measures how many true positive values the model detects and is given by:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

where FN stands for False Negative. False Negatives occur when the model does not detect a object when a object is present. The COCO metric generates three average recall for images containing 0 or 1 objects, 0 to 10 objects and 0 to 100 objects which are called AR@1, AR@10 and AR@100 respectively.

Average Recall/Precision Across Scales give the respective precision and recall scores for images that contain objects of the given sizes:

- Small: objects with $area < 32^2 pixels$
- Medium: objects with $32^2 < area < 96^2 pixels$
- Large: objects with $area > 96^2 pixels$

Table 3 summarizes the model's score for precision and table 4 summarizes its recall. The model has a mAP (the average of all the given precision scores) of 61%. The AR@100 value is used to report the recall since it includes the recall score for all images with 0 to 100 images. The model's recall is at 77%. Note that the model is able to detect and localize large objects at ease, since both the recall and precision values are higher than for medium sized objects. The value for small objects is -1 for both precision and recall, either because the test set does not contain any small sized objects or because none were detected. Furthermore, the recall for images with 1 or 0 objects is lower than the recall for images with 0 to 10 and 0 to 100 objects. This is probably because the images in the data set usually contained more than one object. An indication for this is that the data set contained around 12,500 images which are in aggregate 24,332 labels, meaning that there are around 2 labels/objects per image. Moreover, the average precision score where the IoU threshold is 0.5 is 10% greater than when the threshold is 0.75. So in 10% of the predictions, the overlap between the ground truth bounding boxes and the predicted bounding boxes is between smaller than 0.75 but greater than 0.5.

	Average Precision (AP)			AP Across Scales		
	mAP	AP 0.5 IoU	AP 0.75 IoU	Small	Medium	Large
Precision	0.6123	0.7853	0.6827	-1	0.4146	0.6173

Table 3: Object detector's precision scores using the COCO evaluation metric

	Average Recall (AR)			AR @ 100 Across Scales		
	AR @100	AR @10	AR @1	Small	Medium	Large
Recall	0.7754	0.7716	0.6785	-1	0.5941	0.7805

Table 4: Object detector's recall scores based on the COCO evaluation metric

Overall, a object detector is created that is able to localize and classify kitchen tools with a decent precision and good recall. This is important because object detection is the backbone of this research. The *OperateWith* and *EatWith* edges rely solely on it, while the edges *UsedFor* and *UsedToPrepare* use it in combination with text. The quality of those edges therefore depend on the quality of the object detector. Only the edges *CookedBy* and *Contains* use text and no computer vision.

4 Automatic Knowledge Base Construction

In total there are 4 information extraction engines that generate 7 edges. For each engine, the corresponding section will discuss the commonsense knowledge it aims to extract, i.e. the edge(s), and the assumption about the unstructured data that makes it possible to extract information from it. In addition, the engine assigns weights to the edges it creates, thus a explanation of their assignment is given. For each edge then, relevant results are given and extracted knowledge is evaluated. To do so, the edges are compared to 'ground truth' values, which is knowledge brainstormed by several individuals for each edge in the knowledge base. The comparison determines the number of correct, incorrect and missing predictions, which correspond to the true positive, false positive and false negative values respectively. Given these, the precision (1) and recall (2) scores are calculated which is be the primary metric of evaluation.

4.1 ie_cv_operate_with_engine.py

It is commonsense knowledge that a fork is used in combination with a knife, a whisk is used in a bowl, and a turner is used on a pan. It would not make sense to use a whisk in combination with a blender.

The aim of this engine is to find out exactly that, namely which pair of tools make sense to use in combination with. To this end, it uses the object detector and makes inferences of 449 videos to generate the *OperateWith* (Tool → Tool) edge. The object detector only makes 1 inferences for every second of the video. It could potentially make 20 inferences per second, the maximum frame rate, but doing so would significantly decrease precision since the chance of a false positive are increased by a factor of 20. Furthermore, making 20 inferences per second for 449 videos on a CPU takes approximately 45 days, while making inferences very seconds takes roughly 4 days.

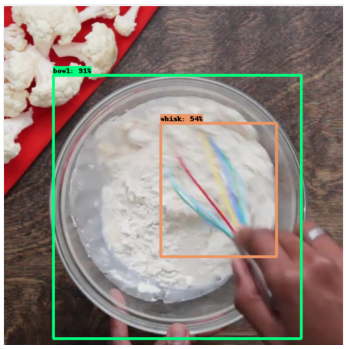

Figure 1: Overlapping frames of a whisk and a bowl

Almost all of the videos are shot in bird eye view. One might infer that two tools are used in combination with each other if their detection frames overlap. For example, Figure 1 gives a snapshot of a video that shows batter being whisked together in a bowl with the correct detection boxes produced by the object detector. Clearly the frames overlap, which the engine will interpret as a whisk and a bowl working well in combination with each other. The tool pairs range across all of the tools subset so a utensil could work in combination with a container, utensil with utensil and container with container.

Weight Assignment

The heuristic is quite simple so it is expected that the engine will create many false edges, i.e. precision will be low. To distinguish between correct and incorrect predictions, the engine also assign weights to the edges. These are determined by (a) in how many videos the two tools interact and (b) in how often it appears in total. The weight for each edge is

given below where a is the number of videos the pair is shown in and b is the total number of occurrences.

$$Weight = 2a + b \tag{3}$$

a is multiplied by two to give it a higher significance than b. This is because it could be that the object detector repeatedly falsely identifies two tools in a video and these two tools are never seen again in another video. Then the weight of the edge between these two tools is low, as it should be.

4.1.1 *OperateWith* (Tool → Tool)

Results & Analysis

In total, the engine identified 268 pair of tools that allegedly work well with each other. Instead of giving all predictions, table 5 and table 6 show the top 5 tool pair with the highest and lowest weights, respectively. As can be seen, the tool pair with the highest weight are all correct predictions while the least significant weights are all incorrect except for the ('measuring-cup', 'blender') pair. In addition, the least significant pair of tools all have a weight of 3 meaning that the weights for incorrect predictions do not range by a lot.

Note that 4 out of 5 most significant tool pairs include a bowl. This confirms that the class imbalance in the object detector's data set 2 is natural since bowls are shown often across many videos. Alternatively, the class imbalance caused these results since circular objects tend to be classified as a bowl. However, this interpretation is unlikely since the predictions are correct, hence the former is true.

Counter	Predicted Tool Pair	Weight	Correct
1	('pinch-bowl', 'bowl')	652	Yes
2	('pan', 'bowl')	595	Yes
3	('bowl', 'plate')	536	Yes
4	('food-container', 'bowl')	410	Yes
5	('turner', 'pan')	398	Yes

Table 5: Top 5 most significant predictions for *OperateWith* edge

Counter	Predicted Tool Pair	Weight	Correct
1	('silicone-spatula', 'lepel')	3	No
2	('lid', 'baking-sheet')	3	No
3	('silicone-spatula', 'cutting-board')	3	No
4	('whisk', 'silicone-spatula')	3	No
5	('measuring-cup', 'blender')	3	Yes

Table 6: Top 5 least significant predictions for *OperateWith* edge

Evaluation

Table 7 summarizes the *OperateWith* evaluation results. As expected, the engine produced many incorrect predictions because the underlying heuristic used is too simple. Just because the frames of two objects overlap does not necessarily mean it is commonsense knowledge to use these two tools in combination. It may mean that they interacted with each other, for example, its plausible to use a small silicone spatula to clean the batter off of a whisk, as was predicted in table 6, but both are not known to work well in combination. Therefore the precision is only at 31.34%.

In comparison, recall is at 64.12% meaning that most of the relevant tool pairs were extracted. This makes sense given that the ground truths only listed 131 tools pairs while the engine found more than double of such pairs. So the information extraction process for the *OperateWith* edge is a little bit like a industrial fishing net since both capture values that they are not needed.

The edges weight are meaningful as the average weight for incorrect pairs are 3 times lower than the weight of correct pairs. This means that although the heuristic upon which the engine operates on is very simple, it is possible to filter out true predictions if the edges weight is above a given threshold.

To conclude, the *OperateWith* edges have low precision but high recall because the information extraction process identifies more pairs than are true. To compensate for it, the edges weight are meaningful and can be used to filter out correct from incorrect predictions.

	Precision	Recall	Avg. Weight of TP	Avg. Weight of FP
Values	31.34%	64.12%	214.62	59.25

Table 7: Evaluation results for *OperateWith* edge

4.2 ie_cv_eat_with_engine.py

It is commonsense to eat a soup with a spoon, a steak with a fork and knife, a burger by hand, and sushi with chopsticks. This is the type of commonsense knowledge this engine aims to extract and represent via the *EatWith (Cutlery → Dish)* edge. It parses the 340 videos associated to 1 recipe text and determines which cutlery is used to eat the dish, which is given by the recipe's title. To extract this knowledge, the object detector analyzes the last 14 seconds of each video since the videos usually end with the meal being served and a closeup shot of the corresponding cutlery taking the food. The object detector uses the maximum frame rate of the videos and makes 20 inferences per second because 1 inference per second only yields 14 predictions which is too little.

The engine uses both the custom made object detector and the SSD ResNet50 V1 FPN 640x640 model made available by the TensorFlow API. It is necessary to use the model pre-trained on the COCO data set because the custom object detector is incapable of detecting hands, which is a common way of eating food. Technically however, the COCO data set does not have 'hands' in its data set either, however it does have 'person'. Since hands are part of a person, the pre-trained model is able to detect hands. Hence, when the results are reported and evaluated, person is used since it is the true class but hands

are meant. The engine still uses the custom object detector because the cutlery in its data set is customized to the videos.

The cutlery with the highest precision is chosen to be the cutlery with which the meal is eaten. The engine makes sure two cutlery's are given for one meal if both their accuracy are both above 55%, otherwise only 1 or no cutlery is predicted. Furthermore, the object detector makes inferences for objects where it is only 47% sure (usually the threshold is put at 50%).

Weight Assignment

Choosing cutlery c base on accuracy is not the only way to determine how to eat the meal. The cutlery could also be determined based on a the one that occurs the most or b the last one detected.

The problem with metric a is that the last 14 seconds also the cutlery to prepare and serve the food. For example, 10 of the last 14 seconds could show a spoon pouring sauce over a steak, and the last 4 seconds show the steak being cut with a fork and a knife. Since the spoon occurred the most, the engine will link the dish with the spoon, although the dish was eaten with a fork and a knife.

The problem with metric b is that the object detector makes mistakes. For example, for the last 14 seconds it correctly identifies the cutlery, but in the last shot of the video it makes a wrong classification. Then the engine will chose the wrong cutlery although it predicted it correctly previously.

However, to use the cutlery with the highest precision is also flawed in many ways, which is why metric a & b are used as weights. The assignment is relatively straight forward:

$$Weight = accuracy(x) + \overbrace{(occurrence(a) + avg.accuracy(a))}^{\text{only if x == a}} + \overbrace{(accuracy(b)}^{\text{only if x == b}} \quad (4)$$

where x is the cutlery with the highest accuracy, a is the cutlery with the highest occurrence, and b is the last seen cutlery. The functions *occurrence()*, *accuracy* and *avg.accuracy* do as their name suggests. Note that $(occurrence(a) + avg.accuracy(a))$ is only added to $accuracy(x)$ if the most occurring cutlery is the same as the highest accuracy, i.e. if $x == a$. The same applies to the addition of $accuracy(b)$. So for example, if $x == fork$, $a ==$ *spoon* and $b ==$ *chopsticks*, then $Weight = accuracy(x)$. However, if $x == fork$, $a ==$ *spoon* and $b == fork$, then $Weight = accuracy(x) + accuracy(b)$ and so on. That way, if the engine identifies different values for the 3 variables, its uncertainty is reflected by the edges weight.

4.2.1 *EatWith* (Cutlery → Dish)

Result & Analysis

For 340 recipes, the engine predicted a total of 408 cutlery. Table 8 displays the most significant *EatWith* edges based on their weight. Similarly, table 9 displays the least significant edges, after having filtered out edges to recipes where nothing was predicted.

Most of the significant edges are correct, i.e. the predicted cutlery corresponds to the cutlery used in the video. All are correct except for one, namely 'The Mediterranean Chicken Curry' is in fact eaten with a fork and not with their hands. Technically however, the engine is not completely incorrect with its prediction. Figure 2 shows a snapshot of the last 14 seconds of the video and the detection frames. The person and the bowl are detected but the barely visible fork is not. This shows the shortcoming of having a object detector that identifies all of a person instead of just the hands. Although the last 14 seconds show the food being consumed with the right cutlery, it is difficult for the engine to identify which one is right because much more is shown.

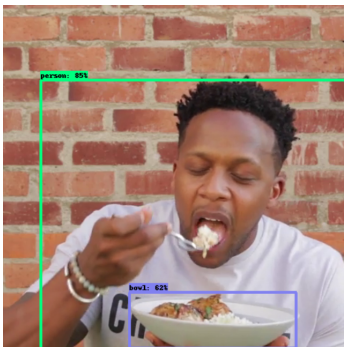

Figure 2: 'Mediterranean Chicken Curry' being eaten

In comparison, almost all of the least significant edges in table 9 are incorrect. It is predicted that the 'Lemon Butter Chicken' dish is eaten with a fork, when really it is eaten with a fork & and knife. The missing knife will be reflected by a lower recall.

Rank	Recipe Name	Predicted Cutlery	Weight	Correct
1	Mexican Sopes	person	398.42	Yes
2	Shahi Paneer	person	376.74	Yes
3	Pakistani-Style Veggie Pakoras	person	363.31	Yes
4	Mediterranean Chicken Curry	person	362.575	No (fork)
5	Fried Chicken	person	348.65	Yes

Table 8: Top 5 most significant predictions for *EatWith* edge

Rank	Recipe Name	Predicted Cutlery	Weight	Correct
1	Gyoza	knife	50	No (person)
2	Lemon Butter Chicken	fork	52	Yes (fork & knife)
3	Roasted Chicken & Veggies	person	52	No (fork)
4	Cheddar Ranch Popcorn Chicken	fork	56	No (person)
5	Vegan Vanilla Ice Cream	knife	57	No (spoon)

Table 9: Top 5 least significant predictions for *EatWith* edge

Evaluation

From the 340 videos, around 100 are used for evaluation. For each dish, the predicted cutlery is compared to the ground truth cutlery. If one of the predicted cutlery match, it is considered a correct prediction, else it is a false prediction. If the ground truth listed more cutlery than were predicted, the missing cutlery in the predictions are registered as false negatives.

The evaluation is split into two metrics, **strict** and **lenient** evaluation because of the following observation. As can be seen in row 1 of table 10, the engine predicts that 17 dishes are eaten with fork & hands and 10 dishes are eaten with a spoon & hands. However, the ground truth values never list hands to be in a combination with any other cutlery because there is no complement to it that would facilitate the consumption of a meal. Given this, if a recipe is eaten with a soup and the engine predicts that a soup & hands are used, both the **strict** and the **lenient** evaluation will classify the spoon as a true prediction. However, the **strict** evaluation will classify the hands as a false prediction. This is technically not wrong since the hand is used to hold the spoon, which is why the **lenient** evaluation does not classify it as a wrong prediction.

Table 11 summarizes the results for both evaluation metrics. As was to be expected, precision is 3.22% higher with the *lenient* evaluation but recall is 8% compared to the *strict* evaluation. So when a dish is eaten with a fork & knife, but the engine predicted fork & person, the *lenient* evaluation removes the person from the pair meaning that knife is considered a false negative, i.e. the engine missed one prediction, hence recall decreases. With the *strict* evaluation, the predicted pair is left as is and person is classified as a false positive, hence precision decreases.

To report the precision and recall of the edge, the average of the two evaluation metrics is used which are both quite high. Recall is obviously higher because it is likely that the object detector identifies cutlery within 14 seconds, but the issue remains whether this is the right cutlery.

The edges weight are again meaningful as the weights of the correct predictions are about 1.2 times higher than of the incorrect ones. Their difference is however not large enough to partition incorrect from correct prediction, as could be done in the *OperateWith*, but there is no need for this since precision is much higher.

Note that the engine predicts some nonsensical cutlery combination like fork & chopsticks as can be seen in table 10. To avoid this, the threshold for predicting two cutlery's could be increased, however that would also decrease the number of sensible cutlery combination like fork & knife which are already low in numbers.

The engine has a false bias towards hands, in both it prediction and weight assignment.

Count	person	fork	spoon	knife	chop-sticks	fork & person	spoon & person	fork & chop-sticks	fork & knife	fork & spoon	knife & person
Prediction	46	11	4	4	0	17	10	1	3	1	1
True	27	43	9	0	7	0	0	0	15	0	0

Table 10: Cutlery and their occurrence count in the videos used for Evaluation.

	Precision	Recall	Avg. Weight of TP	Avg. Weight of FP
Strict	46.33%	90%	197.99	166.57
Lenient	52.77%	82.6%	194.10	153.55
Avg.	49.55%	86.3%	196.05	160.06

Table 11: Evaluation results of *EatWith* edge

Table 10 shows that the engine predicts that more meals are eaten by hand than are true. This may be because hands are more visible than other cutlery, which is often submerged in food, making it harder for the object detector to identify them. Furthermore, all 5 of the most significant predictions are hands, listed in table 8. The last 14 seconds also show a person serving the food, meaning that the most occurring cutlery are also hands. This increases the edges weight.

Overall however, the engine can tell which cutlery to use for which meal with a 49/55% accuracy for around 86 out of 100 recipes. It could be improved by having the object detector identify hands only, and not all parts of a person.

4.3 ie_txt_engine.py

It is commonsense knowledge to cut bread on a cutting board, to poach the egg in a pot and serve food on a plate or bowl. This is the type of knowledge the engine aims to extract and store in the *UsedFor (Container → Verbs & Antonym)*, *CookedBy (Food → Verbs)*, *Contains (Container → ConceptFood & Ingredients)* edges. To do so, the engine parses 2,065 recipes and identifies the food, verbs and containers in the text. Note that no computer vision is used because the text usually mentions the containers explicitly. Computer vision is instead used in the next engine that extracts the same type of knowledge only for utensils.

The engine uses SpaCy's state-of-the-art 'en_core_web_trf' natural language processing model to extract the verbs & nouns in the text. For every verb & noun, the word lemma is taken (e.g. boiling becomes boil) to avoid having separate edges for the same word. Furthermore, note that containers and food are often given as compound nouns, which SpaCy successfully identifies.

For every noun in the sentence, the engine queries ConceptNet to see if it is a food, e.g. for 'chicken', the query will be:
http://api.conceptnet.io/query?start=/c/en/chicken/n/wn/food&rel=/r/IsA. It asks ConceptNet to fetch all nodes that are connected to chicken via a IsA edge. The chicken/n/wn/food part of the query categorizes chicken as food which is vital because without it, ConceptNet returns all IsA edges of the noun unrelated to food. For this query, [poultry] is returned since chicken is a type of poultry. If ConceptNet does not return a node, the engine knows that the noun is not a food.

To identify the food container, every noun is matched to a python dictionary where its keys are the containers represented in the knowledge base and the values are their synonyms. Table 12 shows the containers in the knowledge base and how often the recipes mention them and their synonyms. The engine has a Container class that keeps track of the current container referred to by the recipe, which is used to then establish the

	pan	pot	baking dish	baking sheet	grill	blender	cupcake tin	cutting board	bowl	plate
Count	1,466	584	136	478	81	210	47	111	1,199	166

Table 12: Occurrence of container's and their synonym in 2,065 recipes

UsedFor and *Contains* edge between it and the respective verbs & foods mentioned in the sentence. At the end of each sentence, the list of verbs and food mentioned are used to create the *CookedBy* edge. Note that the engine sets the current container variable in the Container class to None after each step, since the recipe sometimes refers to container different container implicitly.

Weight Assignment

The weight assignment is the same for each of the 3 edges and is simply the count of how many times the edge x is occurs:

$$Weight = occurrence(x) \tag{5}$$

4.3.1 *Contains* (Container → Food Concepts)

Result & Analysis

For 11 container, the engine produced a total of 1,856 edges to ingredients and 893 edges to conceptFoods. Less edges are created to conceptFoods than to ingredients, because several ingredients are mapped to the same conceptFood. Table 13 displays the heaviest *UsedFor* edged, in order from left to right.

Creating edges based on the current container tracked and the food mentioned in the sentence is very coarse grained information extraction procedure and incorrect associations are bound to occur, e.g. it is not expected to cook a tortilla on a grill. However, also some sensible connection are extracted, e.g. muffin is baked in a cupcake tin.

The word sense of food in ConceptNet is broad so some nouns linked to the containers are not really food but only associations of it, e.g. plate has a edge to itself because ConceptNet thinks 'plate is a type of cut of beef' and 'plate is a type of entree'. However, this edge should not exists because plate is not a food. Furthermore, ConceptNet also has some very vague conceptFoods, e.g. blender has a edge to foodstuff which is not very specific. Note that these edges are amongst the heaviest because all food mentioned in the text is a type of foodstuff. Thus, the quality of the edges are dependent on ConceptNet.

The weight of the edge primarily indicate the significance of the food instead of the significance of the edge. For example, pot has a heavy edge to chicken but not with soup, simply because the recipes do not mention soup as often as they do chicken, which is highlighted in table 14.

Container	Most significant ConceptFood	Most significant Ingredient
pan	'flavorer', 'food', 'cut', 'dairy product', 'vegetable'	'side', 'butter', 'onion', 'chicken', 'garlic'
pot	'flavorer', 'food', 'liquid', 'nutrient', 'vegetable'	'water', 'onion', 'salt', 'garlic', 'chicken'
baking dish	'nutriment', 'food', 'condiment', 'flavorer', 'solanaceous vegetable'	'dish', 'sauce', 'chicken', 'cup', 'cheese'
baking sheet	'flavorer', 'cut', 'solanaceous vegetable', 'vegetable', 'concoction'	'pepper', 'salt', 'rack', 'side', 'potato'
grill	'cut', 'food', 'dairy product', 'grain', 'pancake'	'side', 'corn', 'cheese', 'rib', 'tortilla'
blender	'flavorer', 'foodstuff', 'dairy product', 'edible fruit', 'food'	'milk', 'pepper', 'salt', 'juice', 'garlic'
cupcake tin	'quick bread', 'punch', 'concoction', 'frozen dessert', 'dairy product'	'muffin', 'cup', 'batter', 'cupcake', 'ice_cream'
cutting board	'fare', 'cut', 'poultry', 'flavorer', 'helping'	'board', 'chicken', 'side', 'steak', 'salt'
bowl	'flavorer', 'foodstuff', 'food', 'dairy product', 'sweetening'	'salt', 'sugar', 'pepper', 'egg', 'flour'
plate	'cut of beef', 'entree', 'cut', 'flavorer', 'food', 'dish'	'plate', 'cake', 'chicken', 'salt', 'side'

Table 13: Top 5 most significant *Contains* edges

Ingredient	salt	pepper	chicken	sauce	garlic	side	onion	dough	egg	water
Count	2014	1643	1362	1204	1115	1045	1027	1025	987	965

Table 14: Top 10 most occurring foods in 2,065 recipes

	Precision	Recall	Avg. Weight of TP	Avg. Weight of FP
pan	54.79%	80%	62.1	39.59
pot	64.41 %	71.7%	26.75	25.9
baking dish	63.88%	47.91%	8.34	12.92
baking sheet	63.46%	68.04%	17.44	11.68
grill	40.48%	51.51%	4.18	2.16
blender	37.93%	82.5%	29.15	11.55
cupcake tin	34.48%	62.5%	4.5	7.57
cutting board	57.89%	53.22%	8.91	9.96
bowl	68.29%	84.84%	90.13	58.7
plate	79.72	45.38	10.89	3.06
average	56.54%	64.76%	26.23	18.31

Table 15: Evaluation of *Contains* edges

Evaluation

The recipes mention a total of 223 different concetFoods, which the ground truth list uses to link to the containers that could contain them. These are used to determine the engine's true positives, false positives and false negatives rate. Table 15 shows the precision and recall scores for each container, and their average, which is 56.54% and is 64.76% respectively.

Some containers contain a range of food, e.g. almost all food could be served on a plate. The recipes do not mention all of them which is why recall its is 45.38%, however the ones mentioned could be served on a plate, which is why precision is 79.72%. This is different for specialized containers like a cupcake tin that primarily contains batter to bake cupcakes. The recipes mention these specific foods but also unrelated ones which is why recall is at 62.5% but precision is at 34.48%. Thus specialized containers usually have high recall but low precision, while the opposite is true for general purpose containers.

Although the weights depend on the significance of the food itself, they seem to be somewhat meaningful. The average weight of correct edges is 26.23 while the average weight of incorrect predictions is 18.31. This is not true for all, since for 3 containers the average weight for incorrect predictions is higher than the average weight of incorrect predictions.

4.3.2 *UsedFor* (Container → Verbs & Antonyms)

Result & Analysis

From 2,065, the engine produced a total of 1,699 edges to verbs for all container. Table 16 displays the heaviest edges in order from left to right. Most of the verbs associated to the containers seem descriptive, e.g. the most significant verbs linked to cutting board is cut. In addition, the verbs are differentiated, e.g. although both a blender and a bowl are used to mix and combine ingredients, a blender is used to blend and a bowl is used to whisk.

However, not all verbs are descriptive and differentiated, e.g. blender is linked to 'be' (the engine could utilize SpaCy better to filter out such non-descriptive verbs). Moreover, the heaviest edges of pot and pan make it seem as if they are the same tool. Although they are similar and the verbs linked to them are correct, a pot is used to boil while a pan is used to fry. These verbs are not among the heaviest, because just like for the *Contains* edge, the weight depends on the occurrence of the verb and not the significance of the pair, as table 17 shows

Note however that 'bring' is linked to pot which is synonymous to boil. This is because it is used to instruct 'bring to a boil' which appears a total of 102 times. In that sentence, SpaCy correctly identifies 'bring' to be the verb and 'boil' to be the noun, hence boil is not included.

Part of commonsense knowledge is to know what a tool is not used for. To this end, the engine also queries WordNet to produced the antonyms of the extracted verbs since the opposite of a verb should indicate the negated function of the container. Some of the antonyms correctly describe what the container is not used for, e.g. the antonym of mix is segregate, hence a bowl is not used to segregate ingredients.

However, finding the antonym is not the best approach to describe the container's

Container	Most significant Verb	Most significant Antonym
bowl	'combine', 'add', 'mix', 'whisk', 'make'	'take_away', 'subtract', 'segregate', 'unmake', 'break'
pan	'add', 'cook', 'heat', 'remove', 'stir'	'take_away', 'subtract', 'coldness', 'anestrus', 'cool'
pot	'add', 'cook', 'bring', 'stir', 'remove'	'take_away', 'subtract'
baking dish	'pour', 'place', 'transfer', 'spread', 'top', 'remain'	'divest', 'gather', 'fold', 'bottom', 'change'
baking sheet	'line', 'place', 'transfer', 'bake', 'prepare'	'divest'
cutting board	'cut', 'transfer', 'place', 'slice', 'use', 'remove'	'switch_on', 'expand', 'uncut', 'untrimmed', 'unmown', 'divest'
cupcake tin	'grease', 'line', 'fill', 'use', 'place', 'press', 'divide'	'empty', 'divest', 'unite', 'multiply'
blender	'blend', 'add', 'combine', 'mix', 'be'	'take_away', 'subtract', 'segregate', 'differ'
plate	'transfer', 'line', 'drain', 'place', 'serve', 'remove', 'set'	'divest', 'rise'
grill	'place', 'heat', 'cook', 'preheat', 'grill', 'be', 'add'	'divest', 'coldness', 'anestrus', 'cool', 'differ', 'take_away', 'subtract'

Table 16: Top 5 most significant *UsedFor* edges for Containers

Verb	add	cook	enjoy	stir	combine	be	place	remove	mix	serve
Count	4701	2220	2036	1734	1664	1517	1312	1280	1221	1157

Table 17: Top 10 most occurring verbs in 2,065 recipes

Container	Precision	Recall	Avg. Weight of TP	Avg. Weight of FP
bowl	18.18%	95.38%	86.87	9.51
pan	23.24%	92.22%	67.72	11.63
pot	27.63%	85.13%	29.71	6.87
baking dish	31.96%	76.47%	8.53	2.39
baking sheet	31.65%	91.3%	22.41	4.63
cutting board	29.1%	76.2%	6.12	2.82
cupcake tin	28.88%	72.22%	4.38	2.43
blender	22.09%	70.37%	25.21	4.1
plate	20.16%	66.66%	8.37	3.45
grill	31.52%	72.5%	3.75	1.7
average	26.45%	79.85%	26.31	4.94

Table 18: Evaluation of *UsedFor* edges for Containers

negated purpose, because some verbs have no antonym, e.g. 'line', and the antonyms are unrelated to cooking, e.g. a cutting board is not used to unmown. A better approach is to take the complement of the set of verbs linked to a container, i.e. all the verbs in all the recipe that are not associated to the container. However, this approach is not further investigated in the paper.

Evaluation

The edge between the container and the verbs is evaluated by comparing it to the set of verbs listed in the ground truths. As can be seen from table 18, the average precision is 26.45% while average precision is 79.85%. This is because the extracted knowledge lists far more verbs for each container than the ground truths, e.g. the ground truths associates 37 verbs to plate, while the engine listed 119. Thus the engine is bound to make incorrect associations while it includes the most important ones, i.e. recall for plate is 66.66% while precision is only at 20.16%. Far more predictions are made than are needed, thus the analogy of the engine acting like industrial fishing net is again applicable.

The average weight of correct predictions is 26.31 while the average weight of incorrect predictions is 4.94. This appears to be meaningful, however the assignment of the weights is still flawed since they show the significance of the verb and not the pair. Their difference is greater than in the *Contains* edge, probably because the ground truths used common verbs to describe the actions of the containers, which also occur often in the text.

4.3.3 *CookedBy* (Food Concepts → Verbs)

Result

In all the recipes, 220 conceptFood were identified and a total of 744 verbs. The engine produced a total of 8,079 edges between them and table 19 shows some key verbs associated to random conceptFoods. The edges shown are not the heaviest edges because they are not very interesting as they simply list the most occurring verbs. Most edges are sensible, e.g. tenderloin is cut, because the information extraction process suits this edge, i.e. the verbs usually refer to the nouns in the same sentence.

Evaluation

Only 65 out of the 220 conceptFoods were used for evaluation and table 20 shows some of the scores. The average precision of the knowledge extracted is 8.98% while average recall is 61.34%.

One possible explanation for the low precision is that the nodes connected to by this edge both depend on SpaCy & ConceptNet which inevitably make small mistakes in their identification. The Evaluation section will give greater detail to for these values.

The average difference between correct and incorrect predictions is higher than for the other two edges, probably because the weight now depends on the occurrence of both food and verbs mentioned in the text, while the other edges only depended on one.

Overall, the engine successfully produced the *Contains*, *UsedFor* and *CookedBy* edges, with a maP of 30.66% and a average recall of 68.65%. Although the weights primarily indicate the significance of the food or verb in the recipes, it can be concluded that it

Container	Most significant Verbs
soup	'pour', 'simmer', 'ladle', 'boil', 'stir'
butter	'slice', 'whip', 'dip', 'melt', 'dissolve'
saltwater fish	'roast', 'cut', 'slice', 'smoke', 'sear'
sausage	'cook', 'cut', 'brown', 'roast', 'fry'
flour	'flour', 'sprinkle', 'powder', 'coat', 'whisk'
beefsteak	'cut', 'brush', 'flip', 'marinate', 'season'
dough	'flour', 'spread', 'bake', 'roll', 'flatten'
chicken	'turn', 'broil', 'brown', 'fry', 'cook'
tenderloin	'cut', 'massage', 'brush', 'pat', 'pepper'
cake	'bake', 'press', 'decorate', 'sift', 'bake'

Table 19: Some key *CookeBy* edges for random conceptFoods

FoodConcept	Precision	Recall	Avg. Weight of TP	Avg. Weight of TF
soup	12.22%	78.57%	18	2.84
butter	11.32%	54.54%	4.66	1.78
saltwater fish	21.05%	36.36%	2	1.13
sausage	11.62%	71.42%	6.2	2.03
flour	15.62%	55.55%	4.2	1.44
beefsteak	20.45%	60%	4.11	1.49
dough	6.66%	50%	5.14	4.15
chicken	16.66%	33.33%	1.8	2.4
tenderloin	29.16%	46.6%	1.28	1.17
cake	5.74%	100%	11.2	5.18
average	8.98%	61.34%	31.09	5.48

Table 20: Evaluation results of the *CookedBy* edges

is still meaningful since in all of the edges, the average weight of true positives is much higher than the average weight of the false positives.

4.4 ie_multi_modal_engine.py

It is commonsense knowledge to use a turner to flip a patty frying in a pan but use tongs to flip the rib-eye steak searing on the grill. The purpose of this engine is therefore to extract the same type of commonsense knowledge as the ie_txt_engine.py, however specific to utensils rather than containers and store the extracted knowledge in the *UsedFor (Utensils → Verbs)* and the *UsedToPrepare (Utensils → FoodConcepts)* edges.

Unlike containers, the use of utensils are only implied by the verb & food in the text. To make the utensils explicit, the object detector is used to make inferences on the videos and detect the shown utensils. To correctly map the verbs & foods in the text to the utensils shown the in video, the engine needs to parse the video and text synchronously. To do so, the total number of words in the text are divided by the videos duration which gives the words per second rate. This rate determines how many words the engine parses

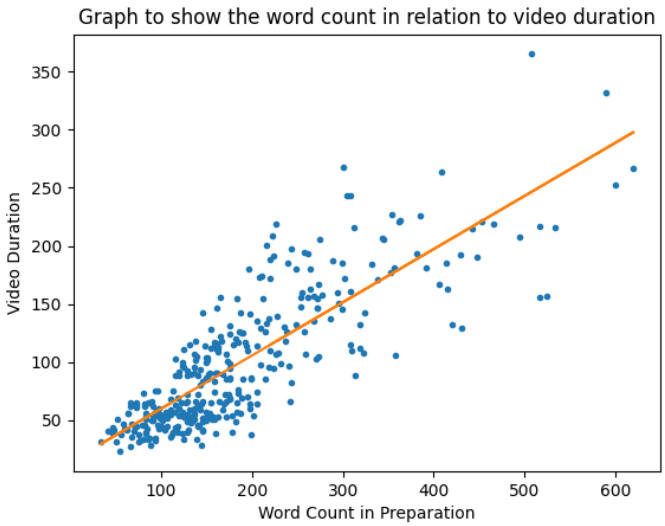

Figure 3: Plot showing for each of the 340 recipes their video duration and its word count

in the text before making a inference on a one second snapshot of the video.

Figure 3 shows for each of the 340 recipes the word count and the video duration. These two variables have correlation coefficient of 0.826, which is also highlighted by the line of best fit in figure 3. It is important for these two variables to be strongly correlated because it means there is a high probability that the utensils shown in the video are used to cook the food mentioned in the text, i.e. the mapping of the utensil to the verbs & foods is correct.

The Sync class in the engine synchronizes the parsing of the two data formats of the recipe. Just like ie_txt_engine.py, the engine also uses ConceptNet to identify the foods and SpaCy to identify the verbs & nouns in the text. At the end of every sentence, the engine links the extracted foods & verbs to the utensils seen in the videos and assigns weights to the pairs.

Weight Assignment

The synchronization scheme is far from perfect and there will always be some discrepancy between what the video shows and what the text says. The weight of the edges should therefore reflect how sure the engine is that the utensil in the video corresponds to the verb & food mentioned in the text.

The video and the text are in sync when both show/refer to the same container. To keep track of the container in the text, the same Container class is used as the ie_txt_engine.py.

However this is not enough to be sure that the verbs & food refer to the utensil, since the video could simply show the utensil laying in the frame without it being used. The engine knows the utensil is used when its detection box overlaps with the detection box of the container. Note that this is the same heuristic used to extract the *OperateWith* edge. Another way of determine whether the utensil is used is by checking if the coordinates of detection box moved between the frames. This increases the complexity of the engine since it needs to keep track of the utensil's coordinates in between frames and is not explored

in this paper.

Therefore, the engine keeps track of 4 counters for each time a utensil is paired to a verb or food. It keeps track of:

- *count*: how often the utensil is paired to a verb/food.
- *sync*: how many times the video was in sync with the text when the pair was established
- *utensil_used*: how many times the utensil overlaps with the current container when the pair is established. For the engine to increment this counter, the video must be in sync with the text, i.e. the container with which the utensil overlaps must be the same container the referred to by the text.
- *accuracy*: the average accuracy of the detected utensil.

Given the definition of the variable counters, the edge's weights are assigned by:

$$Weight = (sync/count) + accuracy + utensil\_used \tag{6}$$

The variable counter that makes up the most weight is therefore *utensil_used* because it measures not only how often the video is in sync with the text, but also how often the utensil is indeed meant. *sync* is divided by *count* to get the fraction to show how often the text was in sync with the video. The more often it was, the closer it is to one. Then the accuracy of the utensil is added to reflect how sure the object detector was with its inference.

Utensils where it does not make sense to use them in combination with a container, such as a icing-spatula, will have a lower weight than utensils where it does make sense, such as a turner. This is because the value of *utensils_used* will probably be close to 0 and the weight depends on the *accuracy* and *sync/count* which never exceed 1. Although it will be lower to other utensils, it will still hopefully show the most significant edges.

4.4.1 *UsedToPrepare* (Utensil → Food Concepts)

Result & Analysis

This edge links food to the utensils used to cook them, hence it is a sibling of the *Contains* edge. The engine produced a total of 1,361 edges for all 23 utensils. Table 21 shows the heaviest edge from left to right of each utensil to the conceptFood and ingredient. Note that amongst them were again vague concepts like foodstuff, which were filtered out as they are not very expressive.

Since the engine extract all the foods that were mentioned in the same sentence that the utensil appeared in, some of the edges only makes sense when they are put into context. For example, a hammer is not used to smash salt, however, it makes sense to tenderize the chicken and then put salt on it. Hence this engine also works as a industrial fishing net.

However, there are also edges that make no sense, e.g. chopsticks are not really used on pancakes. The object detector probably simply misclassified the tool in the frame as chopsticks. The engine is clearly at the mercy of the precision of the object detector.

Utensils	Most significant ConceptFood	Most significant Ingredient
whisk	'spice', 'flavorer', 'sweetening', 'baked goods'	'flour', 'salt', 'sugar', 'cake', 'nutmeg'
mixer	'dairy product', 'sweetening', 'baked goods'	'mixer', 'sugar', 'butter', 'egg', 'salt'
turner	'flavorer', 'vegetable', 'vegetable oil'	'garlic', 'egg', 'onion', 'olive_oil'
silicone-spatula	'flavorer', 'condiment', 'dairy product'	salt', 'garlic', 'pepper', 'egg', 'sauce'
fork	'dairy product', 'appetizer', 'mixed drink'	'coconut', 'cream', 'garnish', 'pancake'
tongs	'flavorer', 'poultry', 'vegetable'	'side', 'salt', 'chicken', 'pepper'
sieve	'spice', 'sweetening', 'dairy product'	'sugar', 'flour', 'egg', 'salt', 'butter'
knife	'sandwich', 'dish', 'grain', 'condiment'	'wrap', 'casserole', 'dish', 'rice'
chopsticks	'pancake', 'vegetable', 'beefsteak'	'side', 'tortilla', 'garlic', 'pepper'
rolling-pin	'edible fruit', 'dairy product', 'juice'	'frosting', 'avocado', 'yogurt'
oven-glove	'flavorer', 'spice', 'sweetening', 'dairy product'	'sugar', 'juice', 'egg', 'ginger'
measuring-cup	'flavorer', 'spice', 'vegetable'	'salt', 'flour', 'pepper', 'garlic'
skimmer	'fish', 'condiment', 'poultry'	'salmon', 'vegetable', 'chicken'
hammer	'baked goods', 'cut', 'flavorer'	'cutlet', 'spinach', 'salt'
spoon	'dairy product', 'beverage', 'sweetening'	'butter', 'salt', 'coconut'
lepel	'dairy product', 'dessert', 'sweetening'	'compote', 'cream_cheese', 'cream'
scoop	'quick bread', 'morsel', 'syrup'	'crumb', 'board', 'cake', 'frosting'
grater	'dairy product', 'root vegetable', 'edible fruit'	'carrot', 'cheese', 'zucchini'
icing-spatula	'dough', 'liquid', 'dish'	'mixer', 'puff', 'pastry', 'center'
masher	'flavorer'	'salt'
ladle	'poultry', 'vegetable', 'flavorer'	'onion', 'chicken', 'steak'
peeler	'solanaceous vegetable', 'root vegetable'	'potato', 'salt', 'vodka', 'cinnamon'
squeezer	'vegetable', 'edible fruit', 'juice'	'meatball', 'lemon_juice', 'lemon'

Table 21: Most significant *UsedToPrepare*

Unlike the *Contains* edge, the heaviest *UsedToPrepare* edge are not always the most occurring foods mentioned in the text, e.g. grater has a heavy edge to zucchini which 92 times in 2,065 recipes. The edge's weight does not depend solely on the occurrence of the pair, but rather how often the food was mentioned when the utensil was used and the video was in sync with the text. Furthermore, the ie_multi_modal_engine.py only parses 340 texts, while the ie_txt_engine.py parsed 2,065 texts, meaning that the most occurring foods added up to be the heaviest *Contains* edge.

Evaluation

Average precision and recall is 36.58% and 54.26% respectively, as table 22 shows. Just like in the *Contains* edge, the scores are computed after comparing the extracted knowledge to ground truths that linked the utensils to the 220 conceptFoods represented in the text.

These scores are lower than for the *Contains* edge because there the containers are simply stated in the text, while the utensils has to be extracted by the object detector which often makes false classifications. In addition, the video and the text may be out

Utensils	Precision	Recall	Avg. Weight of TP	Avg. Weight of FP
whisk	48.48%	87.27%	11.59	4.83
mixer	50.82%	65.96%	20.36	11.84
turner	36.97%	60.27%	4.17	4.15
silicone-spatula	33.88%	87.23%	6.86	7.35
fork	57.53%	43.29%	0.95	0.86
tongs	33.68%	51.61%	9.76	6.46
sieve	25%	72.5%	9.52	5.67
knife	55.31%	23.21%	0.63	0.74
chopsticks	30.76%	34.78%	1.25	1.05
rolling-pin	11.11%	40%	1.31	0.91
oven-glove	4.68%	100%	2.75	3.31
measuring-cup	21.73%	100%	10.47	2.96
skimmer	12.65%	90.91%	1.97	0.91
hammer	11.11%	40%	0.79	0.82
spoon	64.86%	31.58%	6.03	2.83
lepel	65.71%	39.65%	1.95	1.98
scoop	35.29%	50%	0.86	1.16
grater	34.61%	52.94%	1.61	1.02
icing-spatula	34.48%	71.42	0.85	0.93
masher	0%	0%	0	1.25
ladle	53.85%	43.75%	2.21	2.61
peeler	22.22%	18.18%	1.25	0.90
squeezer	45.45%	62.5%	0.54	0.43
average	36.58%	54.26%	4.84	3.02

Table 22: Evaluation of *UsedToPrepare* edge

of sync, so the contents of the video and text may not align. Of course, this problem of synchronization would not exists if the object detector could detect the foods shown in the video and thus establish the edge like that. However that would require a data set for foods which is time & labour intensive.

Regardless, the average weight of correct edges is higher than the average weight of incorrect predictions, which means the weight assignment is significant. Hence, when the text and the video refer to the same container and the utensil is used in combination with that container, then the utensil in the video is indeed used to prepare the foods mentioned in the text.

4.4.2 *UtensilUsedFor* (Utensil → Verbs)

Results & Analysis

This edge is a sibling of the *ContainerUsedFor*, only that it pairs utensils to the verbs. The engine produced a total of 1,933 edges between 23 utensils and the verbs that describe their action. Note that the engine does not find the antonym of the extracted verbs since the bad results obtained for the *ContainerUsedFor* edge makes the attempt futile to

Utensils	Most significant Verb	Edge to its Primary Function
whisk	'whisk', 'add', 'make', 'sift', 'combine'	whisk: yes
mixer	'beat', 'add', 'combine', 'cream', 'incorporate'	mix: yes
turner	'add', 'heat', 'set', 'cook', 'remove'	turn: yes
silicone-spatula	'combine', 'make', 'add', 'grind', 'whisk'	scrape: yes
fork	'continue', 'incorporate', 'pour', 'enjoy', 'keep', 'powder'	eat: no
tongs	'add', 'cook', 'heat', 'be', 'coat'	flip: yes
sieve	'whisk', 'sift', 'stir', 'combine', 'mix'	sift: yes
knife	'drain', 'rinse', 'improve', 'wrap', 'refrigerate', 'layer'	cut: yes
chopsticks	'add', 'cook', 'need', 'flip', 'read'	eat: no
rolling-pin	'spread', 'start', 'work', 'chill', 'place', 'blend', 'sift'	roll: no
oven-glove	'cook', 'whisk', 'combine', 'make', 'begin'	transfer: yes
skimmer	'cook', 'heat', 'add', 'reach', 'scramble'	skim: no
hammer	'remove', 'dry', 'be', 'use', 'roll', 'season'	tenderize: no
spoon	'continue', 'cream', 'make', 'sift', 'powder'	spoon: no
lepel	'cook', 'add', 'be', 'prevent', 'reach'	stir: yes
scoop	repeat', 'remain', 'create', 'dab', 'set', 'have', 'stir'	scoop: yes
grater	'combine', 'whisk', 'shred', 'melt', 'cover'	grate: no
icing-spatula	'beat', 'spread', 'start', 'work', 'remain'	spread: yes
masher	'mix', 'incorporate', 'season'	mash: no
ladle	'add', 'remain', 'sauté', 'cook', 'remove'	ladle: no
peeler	'peel', 'slice', 'sprinkle', 'make', 'combine', 'place'	peel: yes
squeezer	'chop', 'transfer', 'take', 'knead', 'flour', 'cover'	squeeze: no

Table 23: Most significant *UsedFor* edges for Utensils

describe what the utensil is not used for.

Table 23 shows the edge with the heaviest weight from left to right. The engines coarse grained information extraction style which was evident in the results of the *UsedToPrepare* edge is also evident in the results for this edge. For example, the heaviest edge to whisk is whisk itself, which is good, however the the fourth heaviest edge is sift, which is incorrect. The edge was probably established it because the video showed some ingredient being sift into a bowl and were then whisked. Since sift was in the same sentence as the whisk was shown, the engine created a edge between whisk and sift. Hence the engine extracted knowledge that related to whisk and the other tools it works with, but not solely to whisk.

Commonsense knowledge is also defined by the ability to interpret the meaning of a verb based on the context it was written in. However, the engine takes everything literally. For example, the last step of every recipe says 'enjoy!', hence it appears 2036 times in 2065 recipes, as can be seen in table 17. Since it is the last step, the recipe means to say 'eat the meal'. As discussed for the *EatWith* edge, the last video always shows the cutlery with which the meal is eaten, being usually a fork usually. The engine fails to understand the context and establishes a edge to 'enjoy' but has no edge to 'eat' since it is barely mentioned, only 45 times in all 2065 recipes.

Table 23 also indicates whether the utensil has a edge to the verb that defines its primary function. In total, 12 out of the listed 22 utensils do. Thus it is expected that recall will be higher than its precision.

Evaluation

Table 24 shows the precision and recall scores for each utensils and the average weight of true and false positives. Similar to the *ContainerUsedFor* edge, recall is much higher than precision, 15.87% compared to 55.68%, respectively, as the engine associated more links to each utensil than were listed in the ground truths.

The precision and recall scores depend on how applicable the utensil is, e.g. it is more common to use a turner for a recipe than it is to use chopsticks, which is why the engine linked 201 verbs to turner but only 13 to chopsticks. Recall for for turner is therefore 84.21% while its 38.46% for chopsticks. On the other hand, precision for turner is lower than it is for chopsticks. The trade off between precision and recall is evident.

The average weights of correct predictions is twice as large as the average weight of incorrect predictions, meaning that the edge's weight is meaningful again. Their difference is greater then it was for the *UsedToPrepare* edge. This was also the case for their respective sibling edge, i.e. the average difference in weight of correct and incorrect *ContainerUsedFor* edges was 21.82, while their difference for the *Contains* edge was only 7.92. In general, this is because tools are better defined by their actions than by the food they cook. For example, a whisk is always used to whisk (and its synonyms) but a numerous foods can be whisked.

Overall, although the engine had the difficult task of synchronizing the video to the text, it successfully managed to do so and mapped the verbs & foods in the text to the utensils in the video with a relatively good precision and better recall. Although false edges were made, their weight show which edge is correct and which is false.

Utensils	Precision	Recall	Avg. Weight of TP	Avg. Weight of FP
whisk	8.52%	75%	8.51	2.76
mixer	11.38%	77.77%	23.99	4.51
turner	15.92%	84.21%	5.958	3.15
silicone-spatula	11.79%	100%	3.99	2.53
fork	10.91%	60%	1.34	1.26
tongs	17.5%	89.74%	10.81	3.08
sieve	10.52%	80%	5.64	2.63
knife	14.06%	28.12%	1.33	1.08
chopsticks	18.51%	38.46%	2.48	1.65
rolling-pin	11.11%	18.75%	1.32	0.85
oven-glove	11.47%	100%	3.23	1.72
skimmer	14.52%	78.26%	3.976	1.42
hammer	15.38%	40%	1.29	1.17
spoon	16.66%	30.43%	3.42	5.5
lepel	25%	47.61%	1.96	1.68
scoop	10.71	37.5%	1.2	1.42
grater	31.81%	63.63%	2.04	1.97
icing-spatula	14.89%	70%	1.57	1.29
masher	0%	0%	0	1.5
ladle	36.36%	50%	2.97	2.49
peeler	28.57%	40%	1.77	0.97
squeezer	5.88%	10%	0.64	1.41
average	15.84%	55.68%	4	2.06

Table 24: Evaluation of *UsedFor* edge for Utensils

4.5 Properties of the Knowledge Base

If the tool operates on a food, and the foods preparation is described by a set of verbs, then the these sets of verbs should also be used to describe the actions of the tool, i.e. the relation between tools, food and verbs is transitive as figure 4 shows. For all utensils, 67.58% of the verbs used to describe the utensil are also used to describe the food the utensil operates on. This should be the case and validates the knowledge extracted, aside from the precision and recall scores. For example, if a 'knife' has a edge to the food 'bread', and the 'bread' has a edge to the verb 'cut', then 'knife' should also have a verb to 'cut'. In fact, 72% of the verbs used to describe knife also describe the food it operates on. In that sense, the *UsedFor* edge are not needed because the verbs linked to the food also can also be used to describe the actions of the tools.

It would also be interesting to see which tools are synonymous to each other, based on the foods and verbs they have in common. However, this is not further explored in this paper.

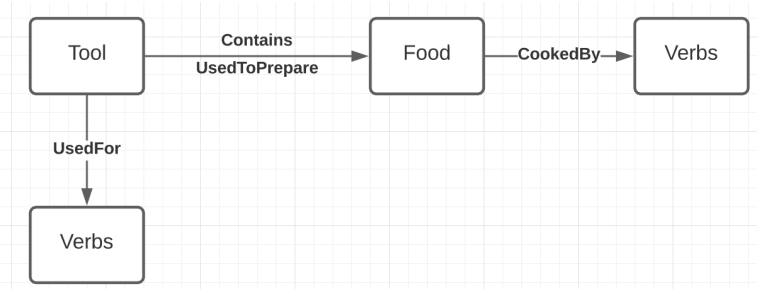

Figure 4: Transitivity of the nodes in the knowledge base

5 Evaluation

All edges were evaluated by comparing the extracted knowledge to manually created ground truths to determine the true & false positives and the false negatives, which were then used to calculate the precision and recall scores. The problem with this evaluation is that the scores depend on the quality of the ground truths.

The ground truths for the *OperateWith* and *EatWith* edge were easy to create because the it involved simply listing pairs of tools that work well with each other and looking at the last 14 seconds of the video to determine which cutlery is used to eat the recipe with, respectively. Thus the precision and recall scores for the two edges are reliable and accurately evaluate the knowledge.

However, it was much more difficult to create the ground truths for the *UsedFor*, *UsedToPrepare*, *Contains* & *CookedBy* edges. It involved pairing tools to food and verbs (and foods to verbs for the latter edge) which requires a lot of imagination and an extensive vocabulary, meaning that the precision and recall scores for each edge is not as reliable. For example, the ground truths only associated 10 verbs to cake, as part of the evaluation of the *CookedBy* edge, while the engine produced 174 edges. It was concluded that the engine is coarse grained, since more verbs are associated than are true. However, it could also be that the 174 verbs accurately describe how to bake a cake but the ground truths simply did not have the vocabulary nor the imagination to envision them.

To potentially compensate for this, WordNet could be queried to find all the synonyms of the verbs in the ground truths. However, this makes the list of truthful verbs associated to the conceptFood explode, e.g. verbs associated to cake increases from 10 to 203. With this list of synonyms, precision marginally increased from 5.74% to 9.19%, while recall drastically decreased from 100% to 7.96%. The problem with this approach is therefore that the synonyms retrieved from WordNet is not specific to cooking and the trade off between precision and recall is too great.

Furthermore, part of the evaluation for the *Contains* edge and its sibling *UsedToPrepare* edge was to associate conceptFoods to tools, and the ground truths did this by using the same 220 conceptFoods represented in the texts. ConceptNet [7] may have way more concepts related to food which were not used for the ease of evaluation. If they were, all scores would be lower.

6 Conclusion

To conclude, this paper introduced 4 information extraction engines that produce commonsense knowledge specific to cooking recipes and represent them via 7 edges, 5 of which are unique. To do so, it trained a object detector on 44 kitchen tools to make inference on recipe videos retrieved from tasty.co/, and used the state-of-the-art natural language processor and existing commonsense knowledge bases to extract knowledge the recipes also retrieved from tasty.co/. The average precision and recall across all edges is 32.18% & 66.62% respectively. The evaluation will discuss the reliability of these scores.

The low precision but relatively high recall highlights the coarse grained information extract process which almost all engines have in common. The analogy to a industrial fishing net is made throughout since far more edges are established than are required. This is especially true for both *UsedFor* edges, e.g. the engine established a total of 1,933 *UtensilUsedFor* edges while the ground truths only identified 437 edges. The ie_cv_operate_with.py engine also identified more than twice as many *OperateWith* edge than the ground truths did. Thus it makes sense for the precision for these edges to be low, as the majority will be incorrect. However it also means recall is high.

Tools are used and designed for a given task, so the verbs associated to them are finite. This is not true for the foods they operate on, which could be infinite. Thus, although the *UtensilUsedFor* edge and the *UsedToPrepare* edge are created by the same ie_multi_modal_engine.py, the information extraction process is far more coarse grained for the former edge than it is for the latter edge. For example, the engine produced 1,361 *UsedToPrepare* edges and the ground truths listed similarly 1,000 *UsedToPrepare* edges, which is reflected in a much higher precision than the *UtensilUsedFor* edge. This comes to show that the information extraction process of the engines is only coarse-grained if the type of commonsense knowledge is specific and the associations are finite.

The precision never exceeds 60% for any edge because the knowledge has to go through some filters in order to extract it, namely the filters of the technologies used. This is especially true for the object detector that has only has a precision of 61.23% and recall of 77.54%, meaning that 40% of the detected objects are incorrect and 23% of the object that could be detected are not. Thus although the heuristics used by the engines about the videos may be correct, the precision of the actual extraction depends on the object detector. This is not so much true for the SpaCy model used, which is really spot on with its sentence analysis. Note that ConceptNet [7] is used in this project to identify food in the sentence, however that is not its primary task so some false classifications will obviously also be made. Thus the information extraction process depends on the technologies used to extract them.

Although the knowledge extracted may often be incorrect, the assignment of the edges weights can be used to separate false from correct predictions. For example, for the *OperateWith* edge the weight of correct predictions was two times higher than the weight of incorrect predictions. The weight assignment of the ie_txt_engine.py was flawed as it did not show the significance of the edge, but depended on the occurrence of the foods & verbs themselves. However, the weights of correct predictions were also higher than incorrect predictions there as well. Thus overall, the weights were meaningful and could be potentially be used to filter out correct from incorrect predictions.

The object detector was the backbone of this research as it made explicit what is implied by the text, making it possible to extract commonsense knowledge. However, creating the data set is notoriously labour intensive and time consuming, thus it would be valuable to research alternative ways to create large data sets, instead of using drawing truth bounding boxes in images manually.

Furthermore, computer vision can be key to general purpose artificial intelligence. However, what was noticeable form the inferences is that the object detector did not regard the context when making inferences. For example, in one frame the object detector correctly identified a fork but in the next frame the fork is a little bit submerged and it identified the same fork as a spoon. It has no notion of memory which would be worthwhile to explore in further research for it to understand the context it is in.

References

[1] Matthew Honnibal and Ines Montani. "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing". To appear. 2017.

[2] Jonathan Huang et al. *Speed/accuracy trade-offs for modern convolutional object detectors*. 2017. arXiv: 1611.10012 [cs.CV].

[3] Dimitrios Kouzis-Loukas. *Learning Scrapy*. Packt Publishing Ltd, 2016.

[4] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[5] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science* (2016), pp. 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[6] George Miller et al. "Introduction to WordNet: An On-line Lexical Database*". In: 3 (Jan. 1991). DOI: 10.1093/ijl/3.4.235.

[7] Robyn Speer, Joshua Chin, and Catherine Havasi. *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge*. 2018. arXiv: 1612.03975 [cs.CL].

7 Appendix

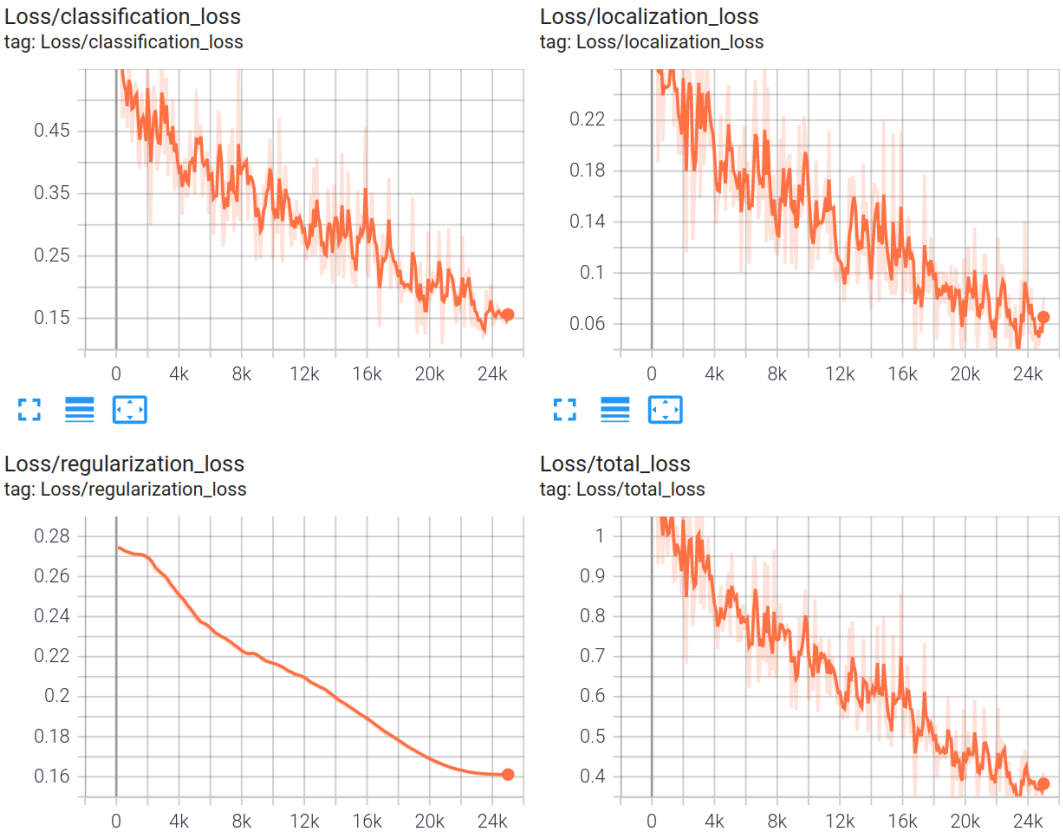

Figure 5: Training progress of the object detector, seen from TensorBoard