Vrije Universiteit Amsterdam

Bachelor Thesis

Leveraging Multi-modal Information Extraction for Automatic Commonsense Knowledge Base Construction specific to Cooking Recipes

---

Author: Leander Finck von Finckenstein (2640632)

1st supervisor:	Dr. Jacopo Urbani
daily supervisor:	Dr. Jacopo Urbani
2nd reader:	Dr. ir. Marc Makkes

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

October 18, 2021

First and foremost, thank you Jacopo
for your mentorship, advice and input throughout the thesis
I value it and learned a lot.

Gratitude and appreciation goes out to
my incredibly kind girlfriend Pauline
for helping me label 12,500 images
and for brainstorming the ground truth values with me.

I am thankful for my beautiful family
who supported me throughout my study.

Abstract

The thesis presents 4 information extraction methods that automatically construct a knowledge base, called the KitchenToolsKB. The knowledge base stores commonsense knowledge found in cooking recipe by pairing kitchen tools to the verbs that describe their actions, to the food they cook, and to other tools that are used in combination. The semantic relations are represented through weighted edges such as *UsedFor*, *EatWith* & *OperateWith* etc., and are extracted from textual and video recipes authored by the tasty.co website. The text and video are parsed simultaneously using a novel synchronization method to pair the tools shown in the videos to the verbs and food mentioned in the text. To identify the tools, the thesis trained a object detector on a custom data set containing 12,500 labelled images of 44 kitchen tools. The evaluation compares the extracted knowledge to a set of ground truth values which yield relatively low precision but high recall scores. All code used for the thesis is available on: github.com/leanderFinck1999/KitchenToolsKB.

1 Introduction

Commonsense is the least common of all senses. While it is debatable whether this phrase is true for people, it is certainly true for computers. If it were not, the livelong sci-fi dream of ever household owning a semi-sentient robot that completes mundane chores would be a reality. Among the mundane household chores to complete is cooking a recipe. Although a recipe is a sequence of instructions, a computer is not be able to follow them because they are rich in commonsense knowledge.

This is a problem for various reasons. Sticking to the sci-fi dream, it is a problem because even if robotic movements are advanced enough to cook a meal, every recipe would need to be translated into a machine readable format. More concretely to reality, it is a problem for search engines that may rank less complex recipes higher than difficult ones. Or it may simply show a featured snippet of the most relevant instruction based on a task-specific query, which it cannot if it does not understand it. The aim of this thesis is therefore to extract commonsense knowledge from recipes and construct the KitchenToolsKB, which a computer can query to follow and understand the preparation of any meal.

Extracting commonsense knowledge is notoriously difficult because it is always implied. For example, if a recipe instructs 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 is the same. For example, a pineapple is peeled with a knife while a apple is peeled with a peeler. Thus commonsense knowledge in cooking is defined by the ability associate the right tool for each instruction, which is difficult to extract since the tools are only implied by the given verbs and foods.

The thesis uses multi-modal information extraction (IE) to extract the commonsense knowledge. It uses a object detector to localize and classify the tools in the recipe videos and pair them to the extracted verbs and foods in the text. The two sources of data are parsed synchronously so that their contents align and the established pairs are correct.

With the KitchenToolsKB, a computer can predict which tool is used for which instruction. This may be useful to a inexperienced cook, as the verbs in the recipe can be annotated with a suggested tools to use. In addition, a list of tools used for the entire recipe can be created, which is useful because the reader can see if they have the necessary

equipment to cook the meal. Furthermore, a search engine may use the created list to advertise the tools to the user. Moreover, the novel way in which the video and text is parsed synchronously can be used for any domain where it is important to map what is shown in the video to the words in the text that accompany it.

The thesis is structured as follows. First, all background information is given, which includes the data collection process from the tasty.co website and a description of all the technologies used to extract the relevant information. Then the related work on the topic is discussed which highlights the novelty of this research, followed by an overview of the KitchenToolsKB and the IE processes that build it. The extracted knowledge is then evaluated based on its precision and recall scores and a use case of the KitchenToolsKB is given. At last, the conclusion summarizes the thesis and gives further research proposals.

2 Background

2.1 Data Collection

The thesis uses Scrapy, a python web-crawling framework, to retrieve the recipes from the tasty.co website, which contains thousands of cooking and baking recipes in both video and text format. The web crawler starts from the tasty.co/ingredient page where all recipes are categorized into their dominant ingredient and listed as hyperlinks in alphabetical order. It follows all hyperlinks that have 'topic', 'compilation', or 'recipe' in their URL and once arrived at a recipe, the crawler parses the HTML and scrapes all data that corresponds to the specified xpaths, including the recipe's preparation, title, video, etc.

Note that the videos are loaded using AJAX meaning that the xpath to the videos URL is not visible to Scrapy as it cannot load it. To compensate for its shortcoming, Selenium is used which deploys a webdriver to open a dummy browser which loads the JavaScript. Thereby the videos appear in the HTML and urllib.request.urlretrieve is used to store the videos locally.

In total, 2,065 textual recipes are scraped and 449 video recipes are retrieved. Out of all the text and videos, there are 1,725 textual recipes that map to 109 videos. This makes sense because loading the video gives a 206 HTTP response code, meaning that only part of the video is loaded. This is a problem because it is impossible to know which text corresponds to which part of the video, meaning that the utensils in the video cannot be paired to the verbs & foods in the text. Recipes that link to unique video URLs and where the word count is smaller than the video duration are identified as recipes where the mapping of text and video is 1 to 1, which leaves 340 recipes.

2.2 Information Extraction from Text

As discussed in the introduction, the verbs and foods mentioned in the text usually imply which tool to use. Hence, they need to be extracted.

2.2.1 Extracting Verbs & Nouns

SpaCy[4], a advanced natural language processing software, is used to identify the verbs & nouns. It offers a variety of state-of-the-art trained models, which differ primarily in speed and accuracy. The thesis uses the 'en_core_web_trf' model as it is the most accurate

(but also the slowest, which does not matter). The model tokenizes every step in every recipe and predicts the most suitable tag for each word[4]. For every verb & noun, the word lemma is taken (e.g. boiling becomes boil) to avoid having separate edges in the knowledge base for the same word.

2.2.2 Extracting Antonyms

For every verb linked to a tool, WordNet[9] is queried to find the verb's antonym. This is to know what a tool is not used for, which the antonym in theory should describe. WordNet is a lexical database that links words by their semantic relations, among them synonymy, hyperonymy, hyponymy, and antonyms[9].

2.2.3 Extracting Food

For every noun extracted, ConceptNet[11], a state-of-the-art commonsense knowledge base, is queried to see if the noun refers to a type of food. For example, if 'chicken' is identified as a noun by SpaCy[4], the query will be:
http://api.conceptnet.io/query?start=/c/en/chicken/n/wn/food&rel=/r/IsA.
It asks ConceptNet[11] 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[11] 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[11] does not return a node, then the noun is not a food. Note that food is often given as compound nouns, like olive oil, which SpaCy successfully identifies. In that case, ConceptNet[11] is queried three times, first for the first noun (olive), then the two nouns combined via an underscore (olive_oil), and lastly the second noun (oil). A separate edge is created for each noun if ConceptNet[11] returns a node.

2.2.4 Extracting Container Tools

Kitchen containers, a subset of kitchen tools, are stated in the text explicitly which table 1 confirms. To identify them, every noun is matched to a predefined python dictionary where the keys are the containers represented in the KitchenToolsKB and the values are their synonyms. The identified container is paired to the foods & verbs mentioned in the same step.

	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 1: Occurrence of container's and their synonym in 2,065 textual recipes

2.3 Information Extraction from Video

Unlike containers, kitchen utensils are always implied by the text. To make the implied utensils explicit and pair them to the verbs and food that define them, computer vision is used. Object detection is the most suitable type of computer vision since the videos show

multiple tools interacting with each other, all of which can to be identified and localized this way.

TensorFlow offers a Object Detection API[6] with models pre-trained on the Common Objects in Context (COCO) data set, which contains thousands of labelled images for 90 common every day objects in context[7]. The objects in the data set classified as kitchen tools are a fork, spoon, knife and bowl[7]. Clearly, the list is not exhaustive and the object detector should be able to identify a number of basic kitchen tools for the KitchenToolsKB to be meaningful.

To this end, a data set needs to be created which consists of images labelled manually with LabelImg[13] of common kitchen tools. Just like the images in the COCO data set, the images in the custom data set should show the tools in context[7] because the object detector may not be able to detect a ladle full with soup if the images it trained on show the ladle being empty. Furthermore, the tools are shown from all possible angles when they are put into context which facilitates their classification and localization. The data set will therefore consists of one second snapshot from some of the recipe videos previously retrieved off of the tasty.co website as they show the tools in context.

2.3.1 Data set statistics

In total, the data set contains around 12,500 labelled images, 24,332 labels, for 44 kitchen tools, which can be downloaded via this this google drive link. Table 2 shows all tools 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. The partition is performed on each class because it maintains the proportion of the classes in the data set. If the entire data set were partitioned into a 9:1 split, it could be that all of the underrepresented tools are in the test set and none in train set, meaning that the object detector is tested on tools it did not train on.

This could be a real scenario since the data set has great class imbalance, e.g. bowl has 4,431 labels while pizza-cutter has 139 labels. To even it out, instead of using 1 frame per second in a video that show a pizza-cutter and other underrepresented objects, 20 frames per second (the maximum frame rate of the videos) is used which marginally increased the number of underrepresented objects in the data set.

2.3.2 Training & Deep Learning used by the Object Detector

To train the custom object detector on the data set, transfer learning is applied on the pre-trained SSD ResNet50 V1 FPN 640x640 model made available by the TensorFlow API[6]. The model uses the state-of-the-art Single Shot MultiBox Detector algorithm (SSD)[8] which makes real-time object detection with an mean average precision (mAP) score of 76.9% on 300x300 sized images[8].

The custom object detector is trained using a batch-size of 8 on a google colab GPU and training ended when the total loss reached a value of about 0.4. The training progress of the individual loss functions can be seen in figure 3 in the appendix and a demo of the object detector can be seen in this video. The trained model can be downloaded here with its corresponding label_map.pbtxt file.

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.

3 Related Work

Knowledge bases are well research as they play a essential role in, e.g. search engines[2], recommendations in social media platforms [12] etc., meaning that numerous already exist. The following compares and contrasts the KitchenToolsKB to other to highlight its novelty.

Both the KitchenToolsKB and ConceptNet[11] represent the knowledge through weighted edges, i.e. triples like ⟨chicken IsA type_of_poultry⟩. This is different to other knowledge bases, like HowToKB, that organizes its knowledge through hierarchical clusters of task frames[2].

However, ConceptNet and the KitchenToolsKB differ in the knowledge base construction as ConceptNet is built from crowed sourced and expert information[11] while the KitchenToolsKB is built automatically through IE processes, like the HowToKB[2] and the WebChild[12]. Although the crowd sourced information is more accurate, it is time consuming and most likely money intensive.

The type of knowledge represented in the KitchenToolsKB is most similar to the HowToKB[2] since both focus on task specific how-to knowledge. The HowToKB taps into online communities like WikiHow to obtain task-solving commonsense knowledge [2] while the KitchenToolsKB obtains commonsense knowledge through cooking recipes retrieved from the tasty.co website which is inherent to solving tasks.

Furthermore, KitchenToolsKB, WebChild[12] and COMET[1] all use machine learning for their IE processes, although the type of machine learning they use is very different. WebChild uses semi-supervised label propagation over graphs to connect nouns to adjectives via edges [12]. In comparison, KitchenToolsKB uses the SSD's single deep neural network[8] to localize and classify tools in videos and a trained convolutional neural network[4] to identify verbs & nouns in the text.

In that regard, the use of computer vision in this research is novel. HowToKB suggested to use it in its further research proposal[2], but did not yet implement it. In addition, the multi-modal extraction methods are also novel since all mentioned knowledge bases extract information from one type of data.

Although the Epic Kitchen data set [3] is not a knowledge base, it is worth mentioning since it and the KitchenToolsKB both created a data set specific to the kitchen. However, the way they created it is different. The images in the KitchenToolsKB data set 2 are labelled manually which is labour intensive and time consuming. In comparison, the images in the Epic Kitchen data set are labelled through narration and a 'pause-and-talk' technique, which makes it possible to have annotations for 100 hours worth of video[3].

4 Overview

The thesis represents commonsense knowledge by connecting the tools to the verbs that describe their action, to the foods they cook, and to the partner tool used in combination. Table 3 shows the list of nodes stored in the knowledge base and the semantic relations between them. The 7 edges are established through 4 IE processes.

The first IE process described in 5.1 receives 449 videos as input and outputs the weighted ⟨Tool, *OperateWith*, Tool⟩ triple. The triple is created for tools whose detection box overlap.

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

Table 3: Nodes & edges contained in the KitchenToolsKB

The second IE process discussed in 5.2 gets the last 14 seconds of 340 recipe videos and its title as input and outputs the weighted ⟨Cutlery, *EatWith*, Dish⟩ triple. The end of the video are used since the final seconds usually show the cutlery consuming the meal.

The third IE process outlined in 5.3 is given 2,065 textual recipes as input and outputs 4 weighted triples, namely ⟨Container, *Contains*, Ingredient&ConceptFood⟩, ⟨ConceptFood, *CookedBy*, Verb⟩, ⟨Container, *UsedFor*, Verbs⟩ and ⟨Container, *NotUsedFor*, Antonyms⟩. The edges are established for each sentence as the verbs describe the actions of the subjects, which usually are the food and the container.

The last IE process explained in 5.4 receives 340 video and text recipes and outputs 3 weighted triples, namely ⟨Utensil, *UsedToPrepare*, Food & ConceptFood⟩, ⟨Utensil, *UsedFor*, Verb⟩, and ⟨Utensil, *NotUsedFor*, Antonym⟩. The video and text are parsed concurrently based on the words per second rate, to correctly map the utensils shown in the video to the verbs and food mentioned in the text.

Each of the corresponding sections delves into greater detail about the relevance of the IE process, the heuristic that drives it, the technologies used for the extraction, and an explanation of how the weights are assigned.

5 Information Extraction Methodology

5.1 IE process 1: *OperateWith*

Relevance: The process creates pairs of tools that work well in combination with each other, as it is commonsense knowledge that a fork is used with a knife, a whisk is used in a bowl, and a turner is used on a pan.

Heuristic: Almost all of the videos are shot in bird eye view, so two tools are used in combination when their detection frames overlap. Figure 1 is a snapshot of a video showing batter being whisked together in a bowl with the correct detection boxes produced by the object detector. Clearly the frames overlap, so the process will create the ⟨whisk, *OperateWith*, bowl⟩ triple. The custom object detector makes 1 inferences for every second

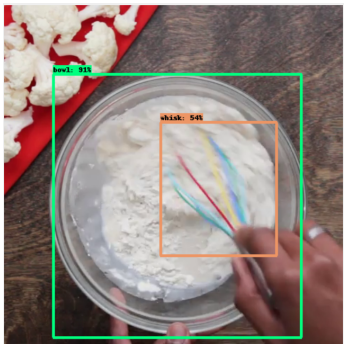

Figure 1: Overlapping frames of a whisk and a bowl

of the videos. It could potentially make 20 inferences per second but doing so would significantly decrease precision since the chance of a false positive are increased by a factor of 20. However, 20 inferences per second would also increase recall.

Weight Assignment: The heuristic is quite simple so false edges are to be expected. To distinguish between correct and incorrect predictions, each edge is assigned a weight, which consists of the *occurrence_in_video*() (in how many videos edge x is seen) and *occurrence*() (total occurrence) of edge x:

$$Weight = 2(occurrence\_in\_video(x)) + occurrence(x) \tag{1}$$

$occurrence\_in\_video(x)$ is multiplied by 2 to give it a higher significance as the validity of the edge increases when the two tools interact across many videos. This reduces the effect of a false classification made by the object detector, as it is unlikely that the same false inference is made across many videos.

5.2 IE process 2: *EatWith*

Relevance: The process connects meals to the cutlery used to eat it, as it is commonsense to eat a steak with a fork and knife, a burger by hand & sushi with chopsticks.

Heuristic: The object detector parses the last 14 seconds of the videos because they usually show the meal being served and a closeup shot of the corresponding cutlery (hands, fork, knife, spoon or chopsticks) taking the food. The object detector makes 20 inferences for each second, since 1 inference per second merely yields 14 predictions. From the $(14 * 20 =)$ 280 inferences, the cutlery with the highest precision is chosen to be the one with which the meal is eaten. Two cutlery's are predicted for one meal if both their precision is above 55%. Furthermore, to increase recall, the object detector makes inferences for objects where it is only 47% sure (usually the threshold is at 50%).

Technologies used: The IE process uses both the custom object detector and the SSD ResNet50 V1 FPN 640x640 model[8]. It is necessary to use the pre-trained model because the custom object detector is incapable of detecting hands. The COCO data set (which the pre-trained model trained on) contains 'person' [7] and since 'hands' are part of a person, the pre-trained model is able to detect hands. The custom object detector also needs to be used because the COCO data set does not contain chopsticks [7], but data set 2 does.

Weight Assignment: The cutlery z with the highest accuracy is not necessarily the one with which the meal is eaten since the last 14 seconds of the video could also show unrelated cutlery, e.g. to serve the dish. A good indication that cutlery z is correct is when it corresponds to the cutlery a that occurs the most and is the last seen cutlery b. The three cutlery's z, a & b make up the edges weight:

$$Weight = accuracy(z) + \overbrace{(occurrence(a) + avg\_accuracy(a))}^{\text{only if z == a}} + \overbrace{accuracy(b)}^{\text{only if z == b}} \tag{2}$$

Thereby, the cutlery z with the highest precision gains weight if the most occurring cutlery a and last seen cutlery b are the same as z. For example, if $z == fork$, $a == spoon$ and $b == chopsticks$, then $Weight = accuracy(z)$. Hence, if different values for the 3 variables are detected, its uncertainty is reflected by a lower weight.

5.3 IE process 3: *ContainerUsedFor*, *ContainerNotUsedFor*, *Contains*, *CookedBy*

Relevance: The IE process connects containers to the foods they cook and to the verbs that describe what it is used for, as it is commonsense knowledge to cut bread on a cutting board and to poach the egg in a pot. It also connects foods to verbs that describe their preparation and containers to the antonyms that describe what it is not used for.

Heuristic: Algorithm 1 shows the rough structure of the IE process. It checks for each word whether it is a verb or food and connects them to the current container, which is assigned on line 5. The pre_parse_to_get_container(sentence) parses the sentence before the verbs and foods are extracted and sets the current container because it may be mentioned after the foods and verbs are, e.g. add the sauce to the pot. If no container is identified, it remains unchanged. Hence it is reset to None on line 2, since the recipe may refer to different containers implicitly in between steps.

Algorithm 1 Overview of the IE process 3

```
1:  for step in 2,065 recipes do
2:      container.current ← None
3:      for sentence in step do
4:          verbs ← []
5:          foods ← []
6:          container.pre_parse_to_assign_container(sentence)
7:          for word in sentence do
8:              if word is verb then                          ▷ SpaCy used
9:                  verbs.append(word)
10:                 container_used_for(container.current, word)
11:                 container_not_used_for(container.current, word_net_antonyms(word))
12:             else if word is noun and word is food then    ▷ SpaCy & ConceptNet
13:                 contains(container.current, word)
14:                 foods.append(word)
15:             end if
16:         end for
17:         cooked_by(foods, verbs)
18:     end for
19: end for
```

Technology used: ConceptNet[11] is queried on line 12 to identify the foods, as discussed in section 2.2.3 and SpaCy[4] is used on line 8 & 12 to extract the verbs & nouns respectively, as discussed in section 2.2.1. For each verb found, WordNet[9] is queried on line 11 to find the verbs antonym, as discussed in section 2.2.2. In addition, the function on line 6 matches each noun in the sentence to the the predefined python dictionary to identify the container, as discussed in section 2.2.4.

Weight Assignment: The weight assignment is the same for each of the 4 edges and is simply the count of how many times the edge x occurs:

$$Weight = occurrence(x) \tag{3}$$

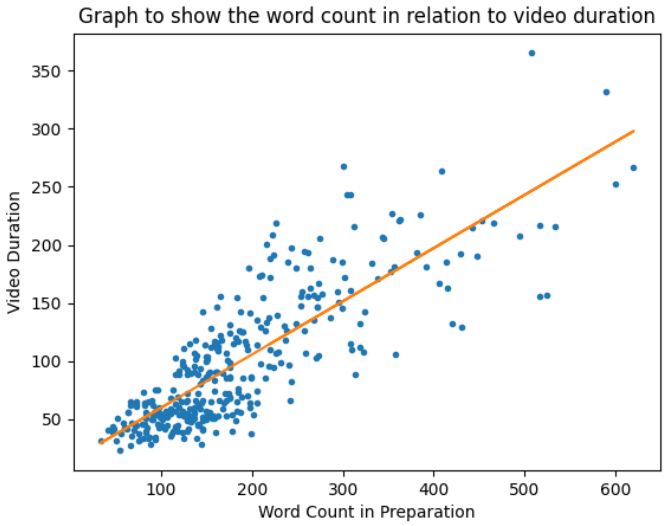

Figure 2: Plot showing for each of the 340 recipes their video duration and its word count

5.4 IE process 4: *UtensilUsedFor*, *UtensilNotUsedFor*, *UsedToPrepare*

Relevance: The type of commonsense knowledge extracted by this IE process is the same as in the previous one, only specific to utensils, as it is commonsense knowledge to flip the patty with a turner, serve the soup with a ladle and mix the batter with a whisk.

Technologies used: The IE process also uses ConceptNet[11], SpaCy[4] and WordNet[9] to find the foods, verbs and antonyms respectively. It follows a similar structure as algorithm 1, however it uses the object detector to fetch the utensils from the videos since unlike containers, utensils are only implied by the given verbs & foods.

Heuristic: To correctly map the verbs & foods in the text to the utensils shown the in video, the two sources need to be parsed synchronously. To do so, the total number of words in the text are divided by the video's duration which gives the words per second rate. This rate determines how many words are parsed before making a inference on a one second snapshot of the video. The word count and the video duration have a correlation coefficient of 0.826 as is shown in the scatter plot in figure 2. It is important for these two variables to be strongly correlated because it means there is a high probability that the words from the text can be mapped to the seconds in the video and can therefore be used to describe what is shown.

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 IE process is that the two sources are in sync, which occurs when both show/refer to the same container. For this, the same Container class from algorithm 1 is used to keep track of the container in the text. In addition, the utensil is meant by the verbs & food if the detection box of the utensil overlaps with the detection box of the container, the same principle used in IE process 5.1. This makes sure that the utensil is indeed used since the video could simply show the utensil laying around. Given the above, each extracted edge x will be given as input to the following functions:

- *occurrence*(): how often the utensil is paired to the verb/food.
- *sync*(): how often the video was in sync with the text when the pair was identified.
- *utensil_used*(): how often the utensil overlaps with the container when the pair is established (the video and text must refer to the same container).
- *avg_accuracy*(): the average accuracy of the detected utensil.

Given the definition of the functions, the edge's weight is the sum of:

$$Weight = (sync(x)/occurrence(x)) + avg\_accuracy(x) + utensil\_used(x) \quad (4)$$

The $utensil\_used(x)$ makes up the most weight because it measures not only how often the video is in sync with the text, but also how often the utensil is indeed meant when edge x was identified. The $avg\_accuracy(x)$ is added to reflect how sure the object detector was with its inference. $sync(x)$ is divided by $occurrence(x)$ as there is no guarantee that the utensil is indeed meant when the edge was created and to avoid double counting $sync(x)$.

6 Limitations

To evaluate the KitchenToolsKB, the edges are compared to ground truth values which is brainstormed commonsense knowledge for each edge. The comparison determines the true positive (TP), false positive (FP) and false negative (FN) values from which the edge's precision 5 and recall 6 scores are calculated. The average precision for all edges is 32.18% while the average recall is 66.62%, which is explained and analyzed in the following sections.

$$Precision = \frac{TP}{TP + FP} \quad (5) \qquad Recall = \frac{TP}{TP + FN} \quad (6)$$

6.1 Evaluating the Object Detector

The evaluation of the object detector helps explain the evaluation results of the IE processes that use it. To evaluate it, the trained model is run on the test set which produces scores for the average recall/precision and the average recall/precision across scales [5] [10], which are summarized in tables 4 and table 5. The average of all the given precision scores is 61.23% while the model's recall is 77.54%. The AP & AR across scales show that the model is better at detecting and localizing large objects than medium or small objects. The value for small objects is -1, either because the test set does not contain any small sized objects or because none were detected. Furthermore, the AR@1 is lower than the AR@10 and AR@100, probably because there are around 2 labels/objects per image in the data set (there are 12,500 images and 24,332 labels). Moreover, for 10% of the predictions, the overlap between the ground truth bounding box and the predicted bounding box is 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 4: Object detector's precision scores

	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 5: Object detector's recall scores

Count	hands	fork	spoon	knife	chop-sticks	fork & hands	spoon & hands	fork & chop-sticks	fork & knife	fork & spoon	knife & hands
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 6: Cutlery and their occurrence count in the videos used for Evaluation.

6.2 Effect of the Object Detector on the IE Processes

The object detector's relatively low precision effects the quality of the edges extracted by IE processes 5.1, 5.2 & 5.4, as they use it to create them.

6.2.1 Comparing IE process 5.3 to 5.4

The edges created by IE process 5.4 (*UsedToPrepare* & *UtensilUsedFor*) have much lower precision scores than of their respective sibling edges (*Contains* & *ContainerUsedFor*) as table 12 shows, because the latter were created by IE process 5.3 which does not use the object detector. However, the lower precision can also be attributed to the synchronization technique used by IE process 5.4 since there is no guarantee that the content of the video corresponds to the content of the text. For example, the *UsedToPrepare* edge would have greater precision if the object detector could detect the food in the video. However, that would require a data set containing labelled images of food which is extremely difficult to create as it would have hundreds of classes for the individual ingredients and meals.

6.2.2 Evaluating IE process 5.2

It is easier for the object detector to detect hands than other cutlery as they are often submerged in food. Since IE process 5.2 chooses the cutlery with the highest precision, more dishes have a *EatWith* edge to hands than are true, as seen in table 6. The object detector's false bias gives rise to two evaluation metrics, **strict** and **lenient**. Table 6 shows that 17 dishes are predicted to be eaten with a fork & hands, however, the ground truth values never pair hands to another cutlery. Thus the **strict** evaluation will classify the hands as incorrect when it is paired, while the **lenient** evaluation does not since technically hands are always used to hold a spoon or a fork. For example, when a dish is eaten with a fork & a knife, but fork & hands are predicted, the **lenient** evaluation removes the hands from the pair and classify knife as a false negative. With the **strict** evaluation, the predicted pair is left as is and hands is classified as a false positive. Hence, the precision of the **lenient** evaluation is 3.22% higher than the precision of the **strict** evaluation, while its recall is 8% lower, as table 8 shows. On average, recall is greater

Dish	Predicted	Weight	Correct
Mexican Sopes	hands	398.42	Yes
Shahi Paneer	hands	376.74	Yes
Veggie Pakoras	hands	363.31	Yes
Chicken Curry	hands	362.575	fork
Fried Chicken	hands	348.65	Yes

(a) The heaviest *EatWith* edge

Dish	Predicted	Weight	Correct
Gyoza	knife	50	hands
Butter Chicken	fork	52	fork&knife
Roasted Chicken	hands	52	fork
Cheddar Ranch	fork	56	hands
Ice Cream	knife	57	spoon

(b) The lightest *EatWith* edge

Table 7: To view all *EatWith* edges, see operate_with.csv

	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 8: Evaluation results of *EatWith* edge

than precision because the object detector is able to identify the cutlery within the last 14 seconds, but the issue remains whether this is the right cutlery.

6.2.3 Evaluating IE process 5.1

Since the object detector has relatively low precision, some of the edges extracted by IE process 5.1 are incorrect, e.g. as seen in table 10, ⟨whisk, *OperateWith*, silicone-spatula⟩ is false. Thus precision for the *OperateWith* edge is only 31.34%, shown in table 9, since it depends on the inferences made by the object detector.

Moreover, its low precision can be attributed to the simple heuristic that drives IE process 5.1 as it extracts many false edges, e.g. the process established 268 *OperateWith* edges, while the ground truth listed 131 edges. Just because the detection frames of two tools overlap does not necessarily mean it is commonsense knowledge that these two tools work well in combination. However, table 9 shows that recall is 64.12%, meaning that although the IE process is coarse-grained, most of the relevant tool pairs were extracted.

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

Table 9: Evaluation results for *OperateWith* edge

Predicted Tool Pair	Weight	Correct
(pinch-bowl, bowl)	652	Yes
(pan, bowl)	595	Yes
(bowl, plate)	536	Yes
(food-container, bowl)	410	Yes
(turner, pan)	398	Yes

(a) The heaviest *OperateWith* edges

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

(b) The lightest *OperateWith* edges

Table 10: To view all edges, see eat_with.csv

6.3 Coarse-grained extraction: The Industrial Fishing net

The simple and coarse-grained IE of process 5.1 is inherent to all processes. As a result, a number of incorrect edges are extracted, which is analogous to the bycatch made by industrial fishing nets. This explains why average precision is so much lower than average recall.

6.3.1 Evaluating *UsedFor* & *CookedBy* edges

The simple design of IE process 5.4 and IE process 5.3 makes the extraction of false edges inevitable. As seen in table 11, IE process 5.4 established the ⟨whisk, *UtensilUsedFor* whisk⟩ edge, which is excellent, however it also created the incorrect ⟨whisk, *UtensilUsedFor* sift⟩ edge. The edges were created probably because sift and whisk appear in the same sentence as the whisk was shown in the video. The unrelated verbs connected to the tool decreases average precision for the *UtensilUsedFor* edge to 15.84% as seen in table

12. Similarly, the IE process 5.3 also produced a lot of bycatch, e.g. the ground truths connected 37 verbs to plate while the IE process extracted 119, hence average precision for the *ContainerUsedFor* edge is 26.45%. However, recall for the *UtensilUsedFor* and *ContainerUsedFor* is 64.76% and 79.85%, respectively as table 12 shows. Thus although the IE processes extracts verbs unrelated to the tool, it also extracts a lot of relevant ones.

Table 13 highlights that the difference between recall and precision is even greater for the *CookedBy* edge. The number of false edges extracted is higher because there is a limited set of words that describe the cooking of a given food. The coarse-grained information extraction style cannot handle such fine-grained knowledge as it is incapable of distinguishing commonsense from nonsensical edges.

6.4 Odd ones Out: Evaluating *Contains* & *UsedToPrepare* edges

The evaluation of some *Contains* & *UsedToPrepare* edges show precision to be greater than recall because the association between tools and food is coarse-grained. For example, almost all food could be served on a plate but the recipes do not mention enough of them, which is why the plate's precision and recall is 79.72% and 45.38%, respectively, as seen in table 12. Similarly, table 12 further shows that the *UsedToPrepare* edges to knife also have greater precision than recall, since the ground truths paired 112 foods to it but IE process 5.4 only extracted 47. Of course, there are other tools that are not used on such a wide range of food, e.g. a cupcake tin primarily bakes cupcakes. These fine-grained edges for specialized tools follow the general trend as the coarse-grained IE processes extract more irrelevant foods than relevant ones. The specialized tools decreases the average precision to 56.54% and 36.58% and increases the average recall to 64.76% and 54.26%, for the *Contains* & *UsedToPrepare* edges respectively, as seen in table 12. However, on average precision and recall are almost equal, hence they are the odd ones out.

6.4.1 ConceptNet's classification of food is broad

In addition to the specialized tools, ConceptNet's[11] false predictions also decreases average precision for the *Contains* & *UsedToPrepare* edge. It's category of food is broad so some tools are connected not directly to food but only associations of it. Table 11 shows plate to have a edge to itself because ConceptNet thinks 'plate is a type of cut of beef' and 'plate is a type of entree'[11]. Thus, the quality of the edges are dependent on ConceptNet[11].

6.5 Reflecting on the *NotUsedFor* edge and the use of Antonyms

Only a few antonyms correctly describe what the tool is not used for, e.g. the antonym of mix is segregate, hence a bowl is not used to segregate ingredients. Overall however, the approach does not describe the tool's negated function because some verbs have no antonym, such as line, and the antonyms retrieved from WordNet[9] are unrelated to cooking, e.g. a cutting board is not used to unmown, as seen in table 11. Note that the *NotUsedFor* edge were therefore evaluated through inspection because it does not make sense to describe what a tool is not used in cooking, when the extracted antonyms/verbs are unrelated to the kitchen. A better approach to describe the negated function of a tool, is by taking the complement of the set of verbs linked to it.

Tool	ConceptFoods	Verbs	Antonyms
pan	flavorer, food, cut, dairy product, vegetable	add, cook, heat, remove, stir	take_away, subtract, coldness, anestrus, 'cool'
pot	flavorer, food, liquid, nutrient, vegetable	add, cook, bring, stir, remove	take_away, subtract, segregate, unbend, end
baking sheet	flavorer, cut, solanaceous vegetable, vegetable, concoction	line, place, transfer, bake, prepare	disarrange, unmake, break, take_away, subtract
cutting board	fare, cut, poultry, flavorer, helping	cut, transfer, place, slice, use, remove	switch_on, expand, uncut, untrimmed, unmown, divest
bowl	flavorer, foodstuff, food, dairy product, sweetening	combine, add, mix, whisk, make	take_away, subtract, segregate, unmake, break
plate	cut of beef, entree, cut, flavorer, food, dish	transfer, line, drain, place, serve, remove, set	unite, multiply, change, divest, gather, fold
whisk	spice, flavorer, sweetening, baked goods, dairy product	whisk, add, make, sift, combine	take_away, subtract, unmake, break, differ
mixer	dairy product, sweetening, baked goods, condiment	beat, add, combine, cream, incorporate	take_away, subtract, disintegrate, unmake, break
turner	flavorer, vegetable, vegetable oil, cut, poultry	add, heat, set, cook, remove	take_away, subtract, coldness, anestrus, cool
silicone-spatula	flavorer, condiment, dairy product, sweetening	combine, make, add, grind, whisk	unmake, break, take_away, subtract, segregate
fork	dairy product, appetizer, meal, cake, tenderloin	continue, incorporate, pour, enjoy, keep, powder	discontinue, disintegrate, suffer, lose, let
tongs	flavorer, poultry, vegetable, cut, drupe	add, cook, heat, be, coat	take_away, subtract, coldness, anestrus, cool
sieve	spice, sweetening, dairy product, punch, drupelet	whisk, sift, stir, combine, mix	segregate, take_away, subtract, divest, differ
knife	sandwich, dish, poultry grain, condiment, fish, pome	drain, rinse, improve, wrap, refrigerate, layer	take_away, go, leave, expand, uncut
chopsticks	cut, pancake, vegetable, beefsteak, winter squash	add, cook, need, flip, read	take_away, subtract, obviate, uncover, blow_up
skimmer	fish, condiment, poultry, cut, alcohol, vegetable oil	cook, heat, add, reach, scramble	coldness, anestrus, cool, take_away, subtract
hammer	baked goods, piece cut, flavorer, vegetable	remove, dry, be, use, roll, season	middle, finish, end, unwind, stand
spoon	dairy product, beverage, sweetening, liquid, flavorer	continue, cream, make, sift, powder	discontinue, unmake, break, disintegrate, empty
lepel	dairy product, punch, sweetening, concoction, vegetable	cook, add, be, prevent, reach	take_away, subtract, differ, let, uncover
scoop	quick bread, morsel, syrup, fare, frozen dessert	repeat, remain, create, dab, set, have, stir	take_away, subtract, rise, change, switch_on
grater	dairy product, root vegetable, edible fruit, summer squash	combine, whisk, shred, melt, cover	take_away, subtract, uncover, switch_on, expand
ladle	poultry, vegetable, flavorer, punch, dish	add, remain, sauté, cook, remove	take_away, subtract, change, rise, unmake
squeezer	vegetable, edible fruit, juice, citrus, flavorer	chop, transfer, take, knead, flour, cover	give, refuse, obviate, abstain, disclaim

Table 11: Some of the heaviest *Contains*, *UsedToPrepare*, *UsedFor* and the resulting *NotUsedFor* edges for some of the tools. To see all edges of all tools can be viewed in the contains.csv, used_to_prepare.csv, container_used_for.csv and utensil_used_for.csv files

Tool	*Contains* & *UsedToPrepare*				*UsedFor*			
	Precision	Recall	Avg. Weight of TP	Avg. Weight of FP	Precision	Recall	Avg. Weight of TP	Avg. Weight of FP
pan	54.79%	80%	62.1	39.59	23.24%	92.22%	67.72	11.63
pot	64.41 %	71.7%	26.75	25.9	27.63%	85.13%	29.71	6.87
baking dish	63.88%	47.91%	8.34	12.92	31.96%	76.47%	8.53	2.39
baking sheet	63.46%	68.04%	17.44	11.68	31.65%	91.3%	22.41	4.63
grill	40.48%	51.51%	4.18	2.16	31.52%	72.5%	3.75	1.7
blender	37.93%	82.5%	29.15	11.55	22.09%	70.37%	25.21	4.1
cupcake tin	34.48%	62.5%	4.5	7.57	28.88%	72.22%	4.38	2.43
cutting board	57.89%	53.22%	8.91	9.96	29.1%	76.2%	6.12	2.82
bowl	68.29%	84.84%	90.13	58.7	18.18%	95.38%	86.87	9.51
plate	79.72	45.38	10.89	3.06	20.16%	66.66%	8.37	3.45
Container average	56.54%	64.76%	26.23	18.31	26.45%	79.85%	26.31	4.94
whisk	48.48%	87.27%	11.59	4.83	8.52%	75%	8.51	2.76
mixer	50.82%	65.96%	20.36	11.84	11.38%	77.77%	23.99	4.51
turner	36.97%	60.27%	4.17	4.15	15.92%	84.21%	5.958	3.15
silicone-spatula	33.88%	87.23%	6.86	7.35	11.79%	100%	3.99	2.53
fork	57.53%	43.29%	0.95	0.86	10.91%	60%	1.34	1.26
tongs	33.68%	51.61%	9.76	6.46	17.5%	89.74%	10.81	3.08
sieve	25%	72.5%	9.52	5.67	10.52%	80%	5.64	2.63
knife	55.31%	23.21%	0.63	0.74	14.06%	28.12%	1.33	1.08
chopsticks	30.76%	34.78%	1.25	1.05	18.51%	38.46%	2.48	1.65
rolling-pin	11.11%	40%	1.31	0.91	11.11%	18.75%	1.32	0.85
oven-glove	4.68%	100%	2.75	3.31	11.47%	100%	3.23	1.72
skimmer	12.65%	90.91%	1.97	0.91	14.52%	78.26%	3.976	1.42
hammer	11.11%	40%	0.79	0.82	15.38%	40%	1.29	1.17
spoon	64.86%	31.58%	6.03	2.83	16.66%	30.43%	3.42	5.5
lepel	65.71%	39.65%	1.95	1.98	25%	47.61%	1.96	1.68
scoop	35.29%	50%	0.86	1.16	10.71	37.5%	1.2	1.42
grater	34.61%	52.94%	1.61	1.02	31.81%	63.63%	2.04	1.97
icing-spatula	34.48%	71.42	0.85	0.93	14.89%	70%	1.57	1.29
masher	0%	0%	0	1.25	0%	0%	0	1.5
ladle	53.85%	43.75%	2.21	2.61	36.36%	50%	2.97	2.49
peeler	22.22%	18.18%	1.25	0.90	28.57%	40%	1.77	0.97
squeezer	45.45%	62.5%	0.54	0.43	5.88%	10%	0.64	1.41
Utensil average	36.58%	54.26%	4.84	3.02	15.84%	55.68%	4	2.06

Table 12: Evaluation results of the *UsedFor Container* & *UsedToPrepare* edges

ConceptFood	Precision	Recall	Avg. Weight of TP	Avg. Weight of TF	Verbs
soup	12.22%	78.57%	18	2.84	pour, simmer, ladle, boil, stir
butter	11.32%	54.54%	4.66	1.78	slice, whip, dip, melt, dissolve
flour	15.62%	55.55%	4.2	1.44	flour, sprinkle, powder, coat, whisk
beefsteak	20.45%	60%	4.11	1.49	cut, brush, flip, marinate, season
cake	5.74%	100%	11.2	5.18	press, decorate, sift, bake
average	8.98%	61.34%	31.09	5.48	

Table 13: Some evaluation results and values of the of the *CookedBy* edge. To view all edges, see food_cooked_by.csv

6.6 Improving Edge's Weight to Filter out Incorrect Edges

Tables 13, 12, 9 and 8 highlight that the edge's weight are meaningful since the average weight of correct edges are higher than the average weight of incorrect edges. Thus the weight could be used to distinguish between correct and incorrect extractions based on a threshold. However, to use the weights as a filter, their assignment in IE process 5.4, 5.2 and 5.3 need to be improved.

Weights produced by IE process 5.4 are lower for edges connected to utensils that are not used in combination with a container, such as a icing-spatula. This is because the value of $utensils\_used(x)$ is close to 0 and the weight depends on the $avg\_accuracy(x)$ and $sync(x)/occurrence(x)$ which never exceeds 1, as seen by the formula 4. To make the weights fair, instead of using the $utensils\_used(x)$, the process could keep track of the utensils coordinates in between frames to determine whether it is used or not.

Furthermore, table 7 shows that the *EatWith* edges to hands are heavier than to other cutlery. This is because hands occur the most during the last 14 seconds of the video as they are usually shown serving the food. Given that IE process 5.2 assigns the weight based on formula 2, this increases their weight and decreases the weight of edges to other cutlery. The weights can be improved by not using the most occurring cutlery as a variable in its assignment.

In addition, the edge's weight assigned by IE process 5.3 is simply the occurrence of the edge (3), which primarily shows the significance of the verbs & food in the text, not the significance of the edge itself. For example, the ⟨pot, *Contains* chicken⟩ edge weights 80 while the ⟨pot, *Contains* soup⟩ only weights 24, simply because chicken is mentioned 1,362 times in all recipes while soup is mentioned 51 times. Ideally, the weights should be swapped since a pot is primarily used to cook soup.

6.7 How true are the ground truths?

It is good to question the validity of the ground truths as they are limited by the imagination and vocabulary of the people who created them, which is especially true for the *UsedFor*, *UsedToPrepare*, *Contains* & *CookedBy* edges. For example, the ground truths associated 10 *CookedBy* edges to cake while IE process 5.3 linked 174 to it. The majority of the edges will be classified as incorrect, which may not yield valid evaluation results since the ground truths could have missed other, exotic verbs which were not envisioned to describe the baking of the cake. The evaluation scores therefore have a margin of error.

To potentially compensate for this, WordNet[9] could be queried to find the synonyms of the verbs listed in the ground truths, which increases the *CookedBy* edges connected to cake from 10 to 203. However, the synonyms are not specific to cooking, since recall drastically decreased from 100% to 7.96% while precision only marginally increased from 5.74% to 9.19%. Hence this approach does not pay off.

Furthermore, the ground truths for *CookedBy* and *EatWith* only used a subset of the foods and dishes to evaluate the edges, hence their scores may be different if a different subset was used. Also the evaluation of the *UsedFor*, *Contains* & *UsedToPrepare* edges used the same 220 conceptFoods extracted by the IE processes to pair them to the tool. There may be many more conceptFoods in ConceptNet[11], which would have decreased precision and recall if they were used.

7 Use Case

After querying the KitchenToolsKB, a computer should know which utensil is implied by the instruction. While parsing the recipe, the client needs to extract the verbs, foods and container mentioned in the sentence (with SpaCy[4], ConceptNet[11] and the predefined python dictionary), and use them as input of the query. The KitchenToolsKB then outputs the utensil connected to the food and verb, based on its *UsedFor*, *UsedToPrepare* and *EatWith* edges. If there are several candidate tools, the *OperateWith* edge is used to filter out the utensils that are not used in the context of the container. Thus, given the verbs, food and container, the KitchenToolsKB should be able to correctly predict the utensil used to complete the instruction.

8 Conclusion

To conclude, the thesis presents the KitchenToolsKB. It is constructed by 4 IE processes that extract 3 main types of semantic relations between kitchen tools and (a) the other tools used in combination, (b) the actions they perform and (c) the food they cook. The first type of relation (a) is represented by the *OperateWith* edge and is created when the detection frames of two tools overlap in IE process 5.1. The second type of relation (b) is represented by the *ContainerUsedFor* and *UtensilUsedFor* edges which are extracted by IE processes 5.3 and 5.4 respectively, while the third type of relation (c) is represented by the *EatWith*, *Contains* and *UsedToPrepare* edges and are extracted by IE processes 5.2, 5.3 and 5.4, respectively. IE process 5.2 uses the object detector to make inferences on the last 14 seconds of the videos as they show the cutlery used to eat the meal. IE process 5.3 uses ConceptNet[11], SpaCy[4] and a predefined python dictionary to extract the foods, verbs and containers respectively, and pair them for each sentence in the recipe. In comparison, IE process 5.4 needs to use the object detector to identify the utensils since unlike the container, utensils are never stated explicitly in the text. To map the utensil in the video to the verbs and foods in the text, the process parses both concurrently based on the words per second rate. Note that IE processes 5.3 and 5.4 also establish the *NotUsedFor* edge, but its results are so poor that it is not worth mentioning as a main edge contained in the KitchenToolsKB. The evaluation of the extracted knowledge shows that all IE processes are quite coarse-grained as many incorrect edges are extracted causing average precision to be only 32.18%. However, all of the edges weight are meaningful and can be used to distinguish between correct and incorrect edges. In addition, the IE processes managed to extract most of commonsense knowledge as average recall is 66.62%.

Future research on the KitchenToolsKB could focus on improving the IE processes. This could be achieved by improving the object detector, either by enlarging the data set or by testing different detection algorithm. Furthermore, the edge's weight assignment could be improved to use them as a filter during the extraction. Future research could also evaluate the KitchenToolsKB's use case, i.e. create precision and recall scores for the suggested tool. Moreover, the synchronization method developed in 5.4 can be used to create a data set similar to [3], but with less effort since the narration performed by the participants to describe what they are doing in the video[3] is replaced by the text from the recipe.

References

[1] Antoine Bosselut et al. *COMET: Commonsense Transformers for Automatic Knowledge Graph Construction*. 2019. arXiv: 1906.05317 [cs.CL].

[2] Cuong Xuan Chu, Niket Tandon, and Gerhard Weikum. "Distilling Task Knowledge from How-To Communities". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 805–814. ISBN: 9781450349130. DOI: 10.1145/3038912.3052715. URL: https://doi.org/10.1145/3038912.3052715.

[3] Dima Damen et al. *The EPIC-KITCHENS Dataset: Collection, Challenges and Baselines*. 2020. arXiv: 2005.00343 [cs.CV].

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

[5] Jan Hosang et al. "What Makes for Effective Detection Proposals?" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.4 (Apr. 2016), pp. 814–830. ISSN: 2160-9292. DOI: 10.1109/tpami.2015.2465908. URL: http://dx.doi.org/10.1109/TPAMI.2015.2465908.

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

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

[8] 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.

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

[10] Hamid Rezatofighi et al. *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression*. 2019. arXiv: 1902.09630 [cs.CV].

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

[12] Niket Tandon et al. "WebChild: Harvesting and Organizing Commonsense Knowledge from the Web". In: *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. WSDM '14. New York, New York, USA: Association for Computing Machinery, 2014, pp. 523–532. ISBN: 9781450323512. DOI: 10.1145/2556195.2556245. URL: https://doi.org/10.1145/2556195.2556245.

[13] Tzutalin. *LabelImg*. Free Software: MIT License. 2015. URL: https://github.com/tzutalin/labelImg.

9 Appendix

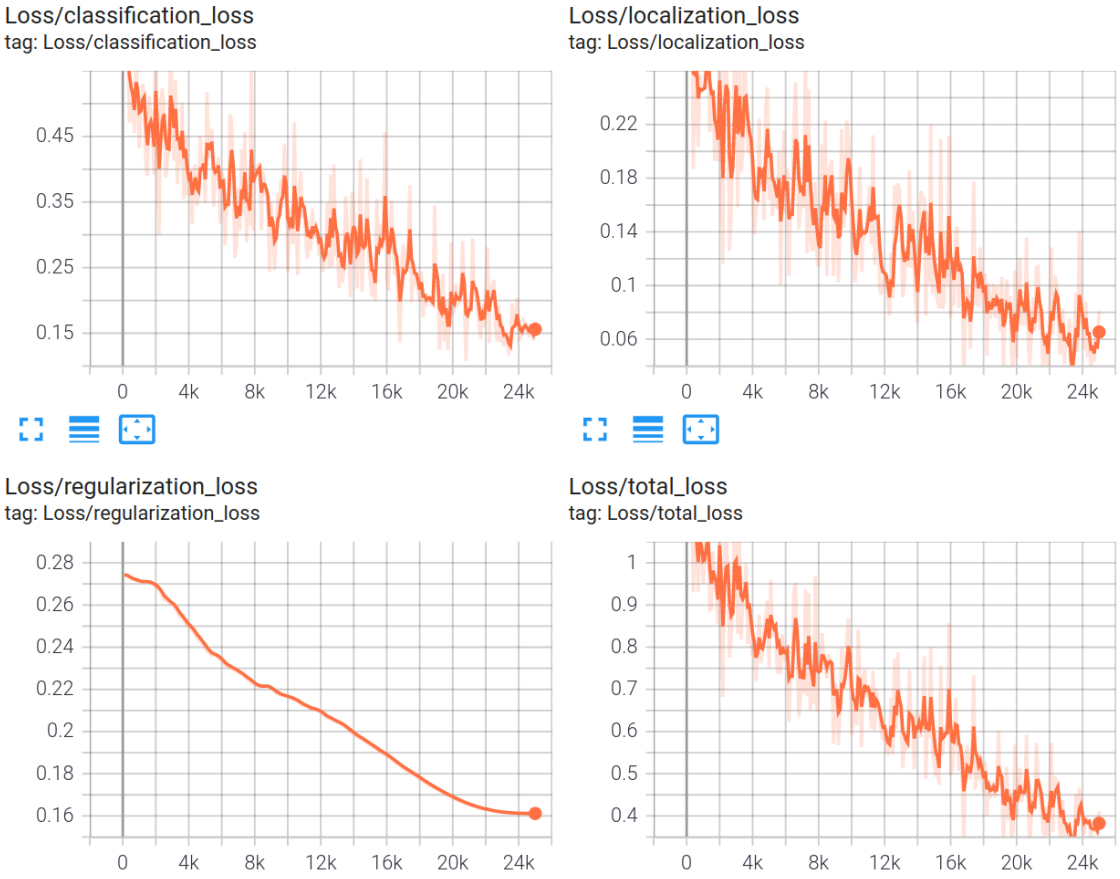

Figure 3: Training progress of the object detector, seen via TensorBoard