What's Cookin'? Interpreting Cooking Videos using Text, Speech and Vision

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston,
Andrew Rabinovich, and Kevin Murphy
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
malmaud@mit.edu
{jonathanhuang, rathodv, nickj, amrabino, kpmurphy}@google.com

Abstract

We present a novel method for aligning a sequence of instructions to a video of someone carrying out a task. In particular, we focus on the cooking domain, where the instructions correspond to the recipe. Our technique relies on an HMM to align the recipe steps to the (automatically generated) speech transcript. We then refine this alignment using a state-of-the-art visual food detector, based on a deep convolutional neural network. We show that our technique outperforms simpler techniques based on keyword spotting. It also enables interesting applications, such as automatically illustrating recipes with keyframes, and searching within a video for events of interest.

1 Introduction

In recent years, there have been many successful attempts to build large "knowledge bases" (KBs), such as NELL (Carlson et al., 2010), KnowItAll (Etzioni et al., 2011), YAGO (Suchanek et al., 2007), and Google's Knowledge Graph/ Vault (Dong et al., 2014). These KBs mostly focus on declarative facts, such as "Barack Obama was born in Hawaii". But human knowledge encompasses procedural information not yet within the scope of such declarative KBs – instructions and demonstrations of how to dance the tango, for example, or how to change a tire on your car. A KB for organizing and retrieving such procedural knowledge could be a valuable resource for helping people (and potentially even robots – e.g., (Saxena et al., 2014; Yang et al., 2015)) learn to perform various tasks.

In contrast to declarative information, procedural knowledge tends to be inherently multimodal. In particular, both language and perceptual information are typically used to parsimoniously describe procedures, as evidenced by the large number of "how-to" videos and illustrated guides on the open web. To automatically construct a multimodal database of procedural knowledge, we thus need tools for extracting information from both textual and visual sources. Crucially, we also need to figure out how these various kinds of information, which often complement and overlap each other, fit together to a form a structured knowledge base of procedures.

As a small step toward the broader goal of aligning language and perception, we focus in this paper on the problem of aligning video depictions of procedures to steps in an accompanying text that corresponds to the procedure. We focus on the cooking domain due to the prevalence of cooking videos on the web and the relative ease of interpreting their recipes as linear sequences of canonical actions. In this domain, the textual source is a user-uploaded recipe attached to the video showing the recipe's execution. The individual steps of procedures are cooking actions like "peel an onion", "slice an onion", etc. However, our techniques can be applied to any domain that has textual instructions and corresponding videos, including videos at sites such as youtube.com, howcast.com, howdini.com or videojug.com.

The approach we take in this paper leverages the fact that the speech signal in instructional videos is often closely related to the actions that the person is performing (which is not true in more general

videos). Thus we first align the instructional steps to the speech signal using an HMM, and then refine this alignment by using a state of the art computer vision system.

In summary, our contributions are as follows. First, we propose a novel system that combines text, speech and vision to perform an alignment between textual instructions and instructional videos. Second, we use our system to create a very large corpus of 180k aligned recipe-video pairs. Third, we derive a large corpus of short video clips, each labeled with a cooking action and a noun phrase. We evaluate the quality of our corpus using human raters, and we show how we can use this corpus to enable within-video search.

2 Data and pre-processing

We first describe how we collected our corpus of recipes and videos, and the pre-processing steps that we run before applying our alignment model. The corpus of recipes, as well as the results of the alignment model, are available for download at `github.com/malmaud/whats_cookin`. [1]

2.1 Collecting a large corpus of cooking videos with recipes

We first searched Youtube for videos which have been automatically tagged with the Freebase mids /m/01mtb (Cooking) and /m/0p57p (recipe), and which have (automatically produced) English-language speech transcripts, which yielded a collection of 7.4M videos. Of these videos, we kept the videos that also had accompanying descriptive text, leaving 6.2M videos.

Sometimes the recipe for a video is included in this text description, but sometimes it is stored on an external site. For example, a video's text description might say "Click here for the recipe". To find the recipe in such cases, we look for sentences in the video description with any of the following keywords: "recipe", "steps", "cook", "procedure", "preparation", "method". If we find any such tokens, we find any URLs that are mentioned in the same sentence, and extract the corresponding document, giving us an additional 206k documents. We then combine the original descriptive text with any additional text that we retrieve in this way.

[1] Approimately 30% of the recipe corpus cannot be released do to copyright considerations.

Class	Precision	Recall	F1
Background	0.97	0.95	0.96
Ingredient	0.93	0.95	0.94
Recipe step	0.94	0.95	0.94

Table 1: Test set performance of text-based recipe classifier.

Finally, in order to extract the recipe from the text description of a video, we trained a classifier that classifies each sentence into 1 of 3 classes: *recipe step*, *recipe ingredient*, or *background*. We keep only the videos which have at least one ingredient sentence and at least one recipe sentence. This last step leaves us with 180,000 videos.

To train the recipe classifier, we need labeled examples, which we obtain by exploiting the fact that many text webpages containing recipes use the machine-readable markup defined at `http://schema.org/Recipe`. From this we extract 500k examples of recipe sentences, and 500k examples of ingredient sentences. We also sample 500k sentences at random from webpages to represent the non-recipe class. Finally, we train a 3-class naïve Bayes model on this data using simple bag-of-words feature vectors. The performance of this model on a separate test set is shown in Table 1.

2.2 Parsing the recipe text

For each recipe, we apply a suite of in-house NLP tools, similar to the Stanford Core NLP pipeline. In particular, we perform POS tagging, entity chunking, and constituency parsing (based on a re-implementation of (Petrov et al., 2006)).[2] Following (Druck and Pang, 2012), we use the parse tree structure to partition each sentence into "micro steps". In particular, we split at any token categorized by the parser as a conjunction only if that token's parent in the sentence's constituency parse is a verb phrase. Any recipe step that is missing a verb is considered noise and discarded.

We then label each recipe step with an optional

[2] Sometimes the parser performs poorly, because the language used in recipes is often full of imperative sentences, such as "Mix the flour", whereas the parser is trained on newswire text. As a simple heuristic for overcoming this, we classify any token at the beginning of a sentence as a verb if it lexically matches a manually-defined list of cooking-related verbs.

action and a list of 0 or more noun chunks. The action label is the lemmatized version of the head verb of the recipe step. We look at all chunked noun entities in the step which are the direct object of the action (either directly or via the preposition "of", as in "Add a cup of flour").

We canonicalize these entities by computing their similarity to the list of ingredients associated with this recipe. If an ingredient is sufficiently similar, that ingredient is added to this step's entity list. Otherwise, the stemmed entity is used. For example, consider the step "Mix tomato sauce and pasta"; if the recipe has a known ingredient called "spaghetti", we would label the action as "mix" and the entities as "tomato sauce" and "spaghetti", because of its high semantic similarity to "pasta". (Semantic similarity is estimated based on Euclidean distance between word embedding vectors computed using the method of (Mikolov et al., 2013) trained on general web text.)

In many cases, the direct object of a transitive verb is elided (not explicitly stated); this is known as the "zero anaphora" problem. For example, the text may say "Add eggs and flour to the bowl. Mix well.". The object of the verb "mix" is clearly the stuff that was just added to the bowl (namely the eggs and flour), although this is not explicitly stated. To handle this, we use a simple recency heuristic, and insert the entities from the previous step to the current step.

2.3 Processing the speech transcript

The output of Youtube's ASR system is a sequence of time-stamped tokens, produced by a standard Viterbi decoding system. We concatenate these tokens into a single long document, and then apply our NLP pipeline to it. Note that, in addition to errors introduced by the ASR system, the NLP system can introduce additional errors, because it does not work well on text that may be ungrammatical and which is entirely devoid of punctuation and sentence boundary markers.

To assess the impact of these combined sources of error, we also collected a much smaller set of 480 cooking videos (with corresponding recipe text) for which the video creator had uploaded a manually curated speech transcript; this has no transcription errors, it contains sentence boundary markers, and it also aligns whole phrases with the video (instead of just single tokens). We applied the same NLP pipeline to these manual transcripts. In the results section, we will see that the accuracy of our end-to-end system is indeed higher when the speech transcript is error-free and well-formed. However, we can still get good results using noisier, automatically produced transcripts.

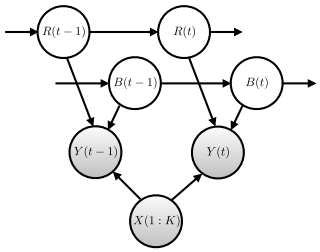


Figure 1: Graphical model representation of the factored HMM. See text for details.

3 Methods

In this section, we describe our system for aligning instructional text and video.

3.1 HMM to align recipe with ASR transcript

We align each step of the recipe to a corresponding sequence of words in the ASR transcript by using the input-output HMM shown in Figure 1. Here $X(1:K)$ represents the textual recipe steps (obtained using the process described in Section 2.2); $Y(1:T)$ represent the ASR tokens (spoken words); $R(t) \in \{1,\ldots,K\}$ is the recipe step number for frame t; and $B(t) \in \{0,1\}$ represents whether timestep t is generated by the background ($B=1$) or foreground model ($B=0$). This background variable is needed since sometimes sequences of spoken words are unrelated to the content of the recipe, especially at the beginning and end of a video.

The conditional probability distributions (CPDs) for the Markov chain is as follows:

$$p(R(t)=r|R(t-1)=r') = \begin{cases} \alpha & \text{if } r=r'+1 \\ 1-\alpha & \text{if } r=r' \\ 0.0 & \text{otherwise} \end{cases}$$

$$p(B(t)=b|B(t-1)=b) = \gamma.$$

This encodes our assumption that the video follows the same ordering as the recipe and that background/foreground tokens tend to cluster together. Obviously these assumptions do not always hold, but they are a reasonable approximation.

For each recipe, we set $\alpha = \frac{|X|}{|Y|}$, the ratio of recipe steps to transcript tokens. This setting corresponds to an *a priori* belief that each recipe step is aligned with the same number of transcript tokens. The parameter γ in our experiments is set by cross-validation to 0.7 based on a small set of manually-labeled recipes.

For the foreground observation model, we generate the observed word from the corresponding recipe step via:

$$\log p(Y(t) = y | R(t) = k, X(1:K), B(t) = 0) \propto \max(\{\text{WordSimilarity}(y, X(i,k)) : i \in X(:,k)\}),$$

where $X(i,k)$ is the i'th word in the k'th recipe step, and WordSimilarity(s,t) is a measure of similarity between words s and t, based on word vector distance.

If this frame is aligned to the background, we generate it from the empirical distribution of words, which is estimated based on pooling all the data:

$$p(Y(t) = y | R(t) = k, B(t) = 1) = \hat{p}(y).$$

Finally, the prior for $p(B(t))$ is uniform, and $p(R(1))$ is set to a delta function on $R(1) = 1$ (i.e., we assume videos start at step 1 of the recipe).

Having defined the model, we "flatten" it to a standard HMM (by taking the cross product of R_t and B_t), then estimate the MAP sequence using the Viterbi algorithm. See Figure 3.4 for an example.

Finally, we can label each segment of the video as follows: use the segmentation induced by the alignment, and extract the action and object from the corresponding recipe step as described in Section 2.2. If the segment was labeled as background by the HMM, we do not apply any label to it.

3.2 Keyword spotting

A simpler approach to labeling video segments is to just search for in the ASR transcript, and then to extract a fixed-sized window around the timestamp where the keyword occurred. We call this approach "keyword spotting". A similar method from (Yu et al., 2014) filters ASR transcripts by part-of-speech tag and finds tokens that match a small vocabulary to create a corpus of video clips (extracted from instructional videos), each labeled with an action/object pair.

In more detail, we manually define a whitelist of $\sim$200 actions (all transitive verbs) of interest, such as "add", "chop", "fry", etc. We then identify when these words are spoken (relying on the POS tags to filter out non-verbs), and extract an 8 second video clip around this timestamp. (Using 2 seconds prior to the action being mentioned, and 6 seconds following.) To extract the object, we take all tokens tagged as "noun" within 5 tokens after the action.

3.3 Hybrid HMM + keyword spotting

We cannot use keyword spotting if the goal is to align instructional text to videos. However, if our goal is just to create a labeled corpus of video clips, keyword spotting is a reasonable approach. Unfortunately, we noticed that the quality of the labels (especially the object labels) generated by keyword spotting was not very high, due to errors in the ASR. On the other hand, we also noticed that the recall of the HMM approach was about 5 times lower than using keyword spotting, and furthermore, that the temporal localization accuracy was sometimes worse.

To get the best of both worlds, we employ the following hybrid technique. We perform keyword spotting for the action in the ASR transcript as before, but use the HMM alignment to infer the corresponding object. To avoid false positives, we only use the output of the HMM if at least half of the recipe steps are aligned by it to the speech transcript; otherwise we back off to the baseline approach of extracting the noun phrase from the ASR transcript in the window after the verb.

3.4 Temporal refinement using vision

In our experiments, we noticed that sometimes the narrator describes an action before actually performing it (this was also noted in (Yu et al., 2014)). To partially combat this problem, we used computer vision to refine candidate video segments as follows. We first trained visual detectors for a large collection of food items (described below). Then, given a candidate video segment annotated with an action/object pair (coming from any of the previous three methods), we find a translation of the window (of up to 3 seconds in either direction) for which the average detector score corresponding to the object is maximized. The intuition is that by detecting when the object in question is visually present in the scene, it is more likely that the corresponding action is actually being performed.

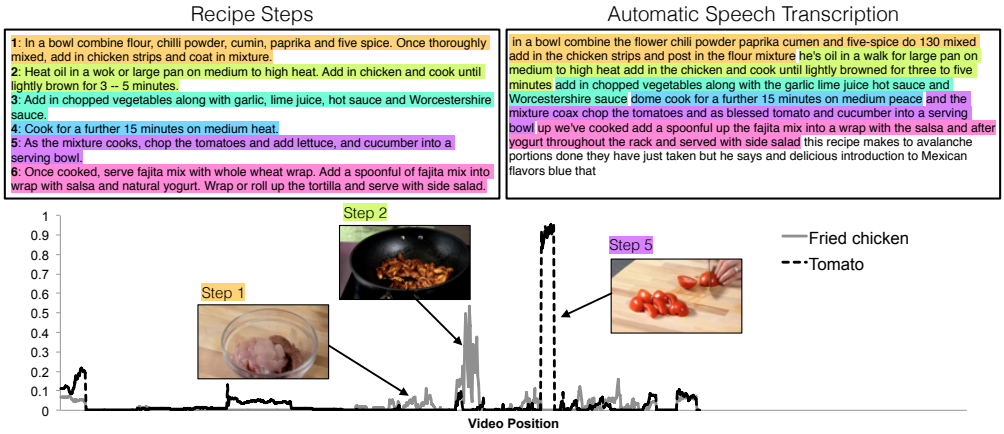


Figure 2: Examples from a Chicken Fajitas recipe at `https://www.youtube.com/watch?v=mGpvZE3udQ4` (figures best viewed in color); Top: Alignment between (left) recipe steps to (right) automatic speech transcript. Tokens from the ASR are allowed to be classified as background steps. Bottom: Detector scores for two ingredients as a function of position in the video.

Training 2800 visual detectors for ingredients. We use a deep convolutional neural network (CNN) to identify food in each frame. Our CNN architecture is similar to the "Inception" model described in (Szegedy et al., 2014) which recently won the Image Net object recognition competition (Russakovsky et al., 2014). To train the model, we constructed a training set consisting of 220 million images harvested from Google Images and Flickr. About 20% of these images contain food, the rest are used to train the background class. In this set, there are 2809 classes of food, including 1005 raw ingredients, such as avocado or beef, and 1804 dishes, such as ratatouille or cheeseburger with bacon. (It is important to note that both of these image sources contain a fair amount of label noise, reaching 50% for some categories.)

Evaluating the classification accuracy of this model on a small holdout set, we achieve an accuracy of 72.3%. To compare the performance of this model to some existing food recognition systems, we also evaluate it on the Food-101 dataset of (Bossard et al., 2014). Despite the fact that we do not use any of their training data, or limit our model to only their 101 classes, we are able to achieve a top-1 classification accuracy of 51.3%, which outperforms their system (based on random forests), which achieves 50.76% accuracy. They mention that a CNN trained on their data can achieve 56.4% accuracy. We believe our CNN would do at least as well as their CNN if trained on their data. However, since this is not the focus of the current paper, we did not retrain our model.

Visual refinement pipeline. For storage and time efficiency, we downsample each video temporally to 5 frames per second and each frame to 224×224 before applying the CNN. Running the food detector on each video then produces a vector of scores (one entry for each of 2809 classes) per timeframe.

There is not a perfect map from the names of ingredients to the names of the detector outputs. For example, an omelette recipe may say "egg", but there are two kinds of visual detectors, one for "scrambled egg" and one for "raw egg". We therefore decided to define the match score between an ingredient and a frame by taking the maximum score for that frame over all detectors whose names matched any of the ingredient tokens (after lemmatization and stopword filtering).

Finally, the match score of a video segment to an object is computed by taking the average score of all frames within that segment. By then scoring and maximizing over all translations of the candidate segment (of up to three seconds away), we produce a final "refined" segment.

3.5 Quantifying confidence via vision and affordances

The output of the keyword spotting and/or HMM systems is an (action, object) label assigned to certain video clips. In order to estimate how much confidence we have in that label (so that we can trade off

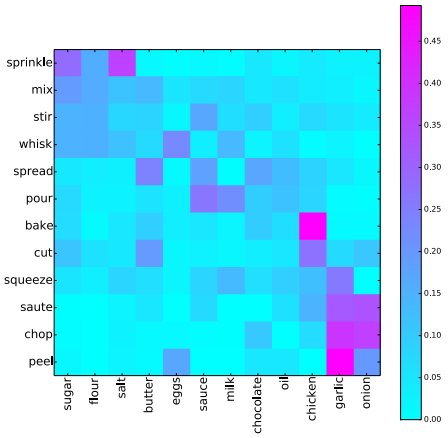


Figure 3: Visualization of affordance model. Entries (a, o) are colored according to $P(\text{object} = o \,|\, \text{action} = a)$.

precision and recall), we use a linear combination of two quantities: (1) the final match score produced by the visual refinement pipeline, which measures the visibility of the object in the given video segment, and (2) an *affordance probability*, measuring the probability that o appears as a direct object of a.

The affordance model allows us to, for example, prioritize a segment labeled as (peel, garlic) over a segment labeled as (peel, sugar). The probabilities $P(\text{object} = o|\text{action} = a)$ are estimated by first forming an inverse document frequency matrix capturing action/object co-occurrences (treating actions as documents). To generalize across actions and objects we form a low-rank approximation to this IDF matrix using a singular value decomposition and set affordance probabilities to be proportional to exponentiated entries of the resulting matrix. Figure 3 visualizes these affordance probabilities for a selected subset of frequently used action/object pairs.

4 Evaluation and applications

In this section, we experimentally evaluate how well our methods work. We then briefly demonstrate some prototype applications.

4.1 Evaluating the clip database

One of the main outcomes of our process is a set of video clips, each of which is labeled with a verb (action) and a noun (object). We generated 3 such labeled corpora, using 3 different methods: keyword spotting ("KW"), the hybrid HMM + keyword spotting ("Hybrid"), and the hybrid system with visual food detector ("visual refinement"). The total number of clips produced by each method is very similar, approximately 1.4 million. The coverage of the clips is approximately 260k unique (action, noun phrase) pairs.

To evaluate the quality of these methods, we created a random subset of 900 clips from each corpus using stratified sampling. That is, we picked an action uniformly at random, and then picked a corresponding object for that action from its support set uniformly at random, and finally picked a clip with that (action, object) label uniformly at random from the clip corpuses produced in Section 3; this ensures the test set is not dominated by frequent actions or objects.

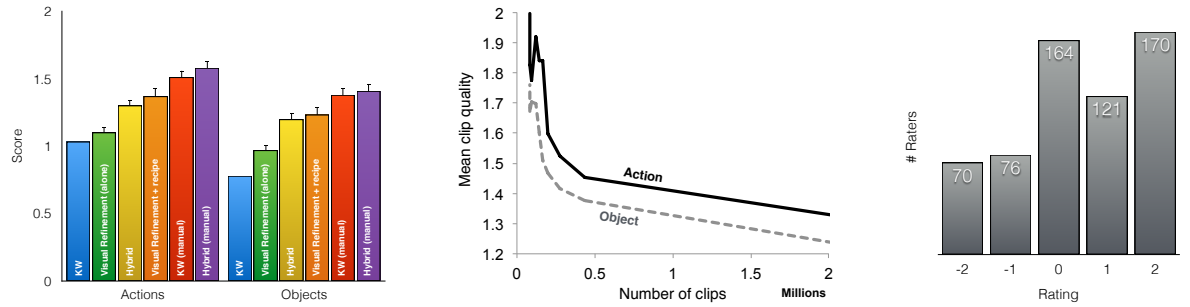
We then performed a Mechanical Turk experiment on each test set. Each clip was shown to 3 raters, and each rater was asked the question "How well does this clip show the given action/object?". Raters then had to answer on a 3-point scale: 0 means "not at all", 1 means "somewhat", and 2 means "very well".

The results are shown in Figure 4(a). We see that the quality of the hybrid method is significantly better than the baseline keyword spotting method, for both actions and objects.[3] While a manually curated speech transcript indeed yields better results (see the bars labeled 'manual'), we observe that automatically generated transcripts allow us to perform almost as well, especially using our alignment model with visual refinement.

Comparing accuracy on actions against that on objects in the same figure, we see that keyword spotting is far more accurate for actions than it is for objects (by over 30%). This disparity is not surprising since keyword spotting searches only for action keywords and relies on a rough heuristic to recover objects. We also see that using alignment (which extracts the object from the "clean" recipe text) and visual refinement (which is trained explicitly to detect ingredients) both help to increase the relative accuracy of objects — under the hybrid method, for example, the accuracy for actions is only 8% better than that of objects.

Note that clips from the HMM and hybrid methods varied in length between 2 and 10 seconds

[3] Inter-rater agreement, measured via Fleiss's kappa by aggregating across all judgment tasks, is .41, which is statistically significant at a $p < .05$ level.



(a) (b) (c)

Figure 4: (a) Clip quality, as assessed by Mechanical Turk experiments on 900 trials. Figure best viewed in color; see text for details; (b) Average clip quality (precision) after filtering out low confidence clips versus # clips retained (recall); (c) Histogram of human ratings comparing recipe steps against ASR descriptions of a video clip. "2" indicate a strong preference for the recipe step; "-2" a strong preference for the transcript. See text for details.

(mean 4.2 seconds), while clips from the keyword spotting method were always exactly 8 seconds. Thus clip length is potentially a confounding factor in the evaluation when comparing the hybrid method to the keyword-spotting method; however, if there is a bias to assign higher ratings to longer clips (which are *a priori* more likely to contain a depiction of a given action than shorter clips), it would benefit the keyword spoting method.

Segment confidence scores (from Section 3.5) can be used to filter out low confidence segments, thus improving the precision of clip retrieval at the cost of recall. Figure 4(b) visualizes this trade-off as we vary our confidence threshold, showing that indeed, segments with higher confidences tend to have the highest quality as judged by our human raters. Moreover, the top 167,000 segments as ranked by our confidence measure have an average rating exceeding 1.75.

We additionally sought to evaluate how well recipe steps from the recipe body could serve as captions for video clips in comparison to the often noisy ASR transript, which serves as a rough proxy for evaluating the quality of the alignment model as well as demonstration a potential application of our method for "cleaning up" noisy ASR captions into complete grammatical sentences. To that end, we randomly selected 200 clips from our corpus that both have an associated action keyword from the transcript as well as a aligned recipe step selected by the HMM alignment model. For each clip, three raters on Mechanical Turk were shown the clip, the text from the recipe step, and a fragment of the ASR transcript (the keyword, plus 5 tokens to the left and right of the keyword). Raters then indicated which description they preferred: 2 indicates a strong preference for the recipe step, 1 a weak preference, 0 indifference, -1 a weak preference for the transcript fragment, and -2 a strong preference. Results are shown in Figure 4(c). Exluding raters who indicated indiffierence, 67% of raters preferred the recipe step.

A potential confound for using this analysis as a proxy for the quality of the alignment model is that the ASR transcript is generally an ungrammatical sentence fragment as opposed to the grammatical recipe steps, which is likely to reduce the raters' approval of ASR captions in the case when both accurately describe the scene. However, if users still on average prefer an ASR sentence fragment which describes the clip correctly versus a full recipe step which is unrelated to the scene, then this experiment still provides evidence of the quality of the alignment model.

4.2 Automatically illustrating a recipe

One useful byproduct of our alignment method is that each recipe step is associated with a segment of the corresponding video.[4] We use a standard keyframe selection algorithm to pick the best frame from each segment. We can then associate this frame with the corresponding recipe step, thus automati-

[4]The HMM may assign multiple non-consecutive regions of the video to the same recipe step (since the background state can turn on and off). In such cases, we just take the "convex hull" of the regions as the interval which corresponds to that step. It is also possible for the HMM not to assign a given step to any interval of the video.

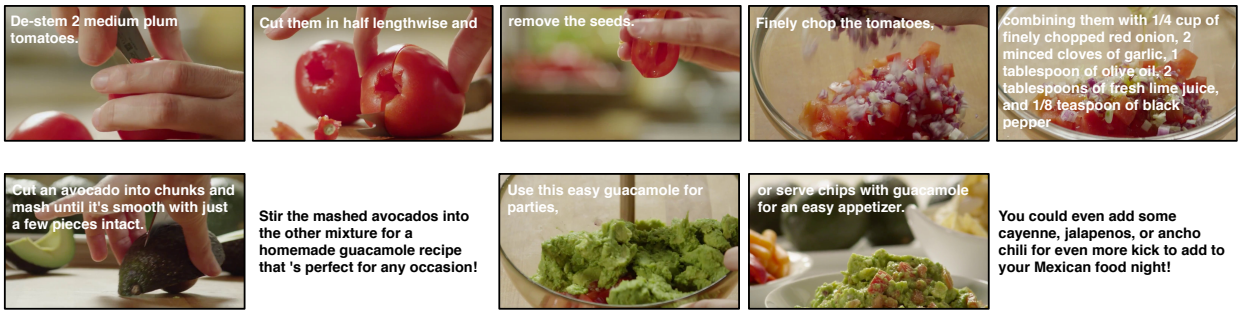


Figure 5: Automatically illustrating a Guacamole recipe from `https://www.youtube.com/watch?v=H7Ne3s2021U`.

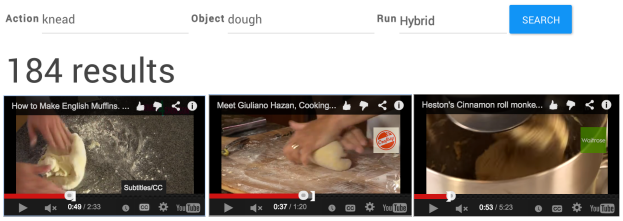


Figure 6: Searching for "knead dough". Note that the videos have automatically been advanced to the relevant frame.

cally illustrating the recipe steps. An illustration of this process is shown in Figure 5.

4.3 Search within a video

Another application which our methods enable is search within a video. For example, if a user would like to find a clip illustrating how to knead dough, we can simply search our corpus of labeled clips, and return a list of matches (ranked by confidence). Since each clip has a corresponding "provenance", we can return the results to the user as a set of videos in which we have automatically "fast forwarded" to the relevant section of the video (see Figure 6 for an example). This stands in contrast to standard video search on Youtube, which returns the whole video, but does (in general) not indicate where within the video the user's search query occurs.

5 Related work

There are several pieces of related work. (Yu et al., 2014) performs keyword spotting in the speech transcript in order to label clips extracted from instructional videos. However, our hybrid approach performs better; the gain is especially significant on automatically generated speech transcripts, as shown in Figure 4(a).

The idea of using an HMM to align instructional steps to a video was also explored in (Naim et al., 2014). However, their conditional model has to generate images, whereas ours just has to generate ASR words, which is an easier task. Furthermore, they only consider 6 videos collected in a controlled lab setting, whereas we consider over 180k videos collected "in the wild".

Another paper that uses HMMs to process recipe text is (Druck and Pang, 2012). They use the HMM to align the steps of a recipe to the comments made by users in an online forum, whereas we align the steps of a recipe to the speech transcript. Also, we use video information, which was not considered in this earlier work.

(Joshi et al., 2006) describes a system to automatically illustrate a text document, however they only generate one image, not a sequence, and their techniques are very different.

There is also a large body of other work on connecting language and vision; we only have space to briefly mention a few key papers. (Rohrbach et al., 2012b) describes the MPII Cooking Composite Activities dataset, which consists of 212 videos collected in the lab of people performing various cooking activities. (This extends the dataset described in their earlier work, (Rohrbach et al., 2012a).) They also describe a method to recognize objects and actions using standard vision features. However, they do not leverage the speech signal, and their dataset is significantly smaller than ours.

(Guadarrama et al., 2013) describes a method for generating subject-verb-object triples given a short video clip, using standard object and action detectors. The technique was extended in (Thomason et al., 2014) to also predict the location/ place. Furthermore, they use a linear-chain CRF to combine the visual scores with a simple (s,v,o,p) language model

(similar to our affordance model). They applied their technique to the dataset in (Chen and Dolan, 2011), which consists of 2000 short video clips, each described with 1-3 sentences. By contrast, we focus on aligning instructional text to the video, and our corpus is significantly larger.

(Yu and Siskind, 2013) describes a technique for estimating the compatibility between a video clip and a sentence, based on relative motion of the objects (which are tracked using HMMs). Their method is tested on 159 video clips, created under carefully controlled conditions. By contrast, we focus on aligning instructional text to the video, and our corpus is significantly larger.

6 Discussion and future work

In this paper, we have presented a novel method for aligning instructional text to videos, leveraging both speech recognition and visual object detection. We have used this to align 180k recipe-video pairs, from which we have extracted a corpus of 1.4M labeled video clips – a small but crucial step toward building a multimodal procedural knowlege base. In the future, we hope to use this labeled corpus to train visual action detectors, which can then be combined with the existing visual object detectors to interpret novel videos. Additionally, we believe that combining visual and linguistic cues may help overcome longstanding challenges to language understanding, such as anaphora resolution and word sense disambiguation.

References

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. 2014. Food-101 – mining discriminative components with random forests. In *ECCV*.

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka Jr., and T. Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *ACL*, HLT '11, pages 190–200, Stroudsburg, PA, USA. Association for Computational Linguistics.

X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proc. of the Int'l Conf. on Knowledge Discovery and Data Mining*.

Gregory Druck and Bo Pang. 2012. Spice it up?: Mining refinements to online instructions from user generated content. In *Proc. ACL*, pages 545–553.

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. 2011. Open Information Extraction: the Second Generation. In *Intl. Joint Conf. on AI*.

S. Guadarrama, N. Krishnamoorthy, G. Malkarnenkar, S. Venugopalan, R. Mooney, T. Darrell, and K. Saenko. 2013. YouTube2Text: Recognizing and describing arbitrary activities using semantic hierarchies and Zero-Shot recognition. In *Intl. Conf. on Computer Vision*, pages 2712–2719.

Dhiraj Joshi, James Z Wang, and Jia Li. 2006. The story picturing Engine-A system for automatic text illustration. *ACM Trans. Multimedia Comp., Comm. and Appl.*, 2(1):1–22.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. http://arxiv.org/abs/1301.3781.

I Naim, Y C Song, Q Liu, H Kautz, J Luo, and D Gildea. 2014. Unsupervised alignment of natural language instructions with video segments. In *AAAI*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

M Rohrbach, S Amin, M Andriluka, and B Schiele. 2012a. A database for fine grained activity detection of cooking activities. In *CVPR*, pages 1194–1201.

Marcus Rohrbach, Michaela Regneri, Mykhaylo Andriluka, Sikandar Amin, Manfred Pinkal, and Bernt Schiele. 2012b. Script data for Attribute-Based recognition of composite activities. In *ECCV*, pages 144–157.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2014. ImageNet Large Scale Visual Recognition Challenge. http://arxiv.org/abs/1409.0575.

Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra K Misra, and Hema S Koppula. 2014. RoboBrain: Large-Scale knowledge engine for robots. http://arxiv.org/pdf/1412.0691.pdf.

F. M. Suchanek, G. Kasneci, and G. Weikum. 2007. YAGO: A Large Ontology from Wikipedia and WordNet. *J. Web Semantics*, 6:203217.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Ra-

binovich. 2014. Going deeper with convolutions. http://arxiv.org/abs/1409.4842.

J Thomason, S Venugopalan, S Guadarrama, K Saenko, and R Mooney. 2014. Integrating language and vision to generate natural language descriptions of videos in the wild. In *Intl. Conf. on Comp. Linguistics*.

Y Yang, Y Li, and Y Aloimonos. 2015. Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web. In *AAAI*.

Haonan Yu and JM Siskind. 2013. Grounded language learning from video described with sentences. In *ACL Conference*.

Shoou-I Yu, Lu Jiang, and Alexander Hauptmann. 2014. Instructional videos for unsupervised harvesting and learning of action examples. In *Intl. Conf. Multimedia*, pages 825–828. ACM.