Spice it Up? Mining Refinements to Online Instructions from User Generated Content

Gregory Druck
Yahoo! Research
gdruck@gmail.com

Bo Pang
Yahoo! Research
bopang42@gmail.com

Abstract

There are a growing number of popular web sites where users submit and review instructions for completing tasks as varied as building a table and baking a pie. In addition to providing their subjective evaluation, reviewers often provide *actionable* refinements. These refinements clarify, correct, improve, or provide alternatives to the original instructions. However, identifying and reading all relevant reviews is a daunting task for a user. In this paper, we propose a generative model that jointly identifies user-proposed refinements in instruction reviews at multiple granularities, and aligns them to the appropriate steps in the original instructions. Labeled data is not readily available for these tasks, so we focus on the unsupervised setting. In experiments in the recipe domain, our model provides 90.1% F_1 for predicting refinements at the review level, and 77.0% F_1 for predicting refinement segments within reviews.

1 Introduction

People turn to the web to seek advice on a wide variety of subjects. An analysis of web search queries posed as questions revealed that "how to" questions are the most popular (Pang and Kumar, 2011). People consult online resources to answer technical questions like "how to put music on my ipod," and to find instructions for tasks like tying a tie and cooking Thanksgiving dinner. Not surprisingly, there are many Web sites dedicated to providing instructions. For instance, on the popular DIY site *instructables.com* ("share what you make"), users post instructions for making a wide variety of objects ranging from bed frames to "The Stirling Engine, absorb energy from candles, coffee, and more![1]" There are also sites like *allrecipes.com* that are dedicated to a specific domain. On these community-based instruction sites, instructions are posted and reviewed by users. For instance, the aforementioned "Stirling engine" has received over 350 reviews on *instructables.com*.

While user-generated instructions greatly increase the variety of instructions available online, they are not necessarily foolproof, or appropriate for all users. For instance, in the case of recipes, a user missing a certain ingredient at home might wonder whether it can be safely omitted; a user who wants to get a slightly different flavor might want to find out what substitutions can be used to achieve that effect. Reviews posted by other users provide a great resource for mining such information. In recipe reviews, users often offer their customized version of the recipe by describing changes they made: e.g., "I halved the salt" or "I used honey instead of sugar." In addition, they may clarify portions of the instructions that are too concise for a novice to follow, or describe changes to the cooking method that result in a better dish. We refer to such *actionable* information as a *refinement*.

Refinements can be quite prevalent in instruction reviews. In a random sample of recipe reviews from *allrecipes.com*, we found that 57.8% contain refinements of the original recipe. However, sifting through all reviews for refinements is a daunting

[1] http://www.instructables.com/id/The-Sterling-Engine-absorb-energy-from-candles-c

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 545–553,
Jeju, Republic of Korea, 8-14 July 2012. ©2012 Association for Computational Linguistics

task for a user. Instead, we would like to automatically identify refinements in reviews, summarize them, and either create an annotated version of the instructions that reflects the collective experience of the community, or, more ambitiously, revise the instructions directly.

In this paper, we take first steps toward these goals by addressing the following tasks: (1) identifying reviews that contain refinements, (2) identifying text segments within reviews that describe refinements, and (3) aligning these refinement segments to steps in the instructions being reviewed (Figure 1 provides an example). Solving these tasks provides a foundation for downstream summarization and semantic analysis, and also suggests intermediate applications. For example, we can use review classification to filter or rank reviews as they are presented to future users, since reviews that contain refinements are more informative than a review which only says "Great recipe, thanks for posting!"

To the best of our knowledge, no previous work has explored this aspect of user-generated text. While review mining has been studied extensively, we differ from previous work in that instead of focusing on *evaluative* information, we focus *actionable* information in the reviews. (See Section 2 for a more detailed discussion.)

There is no existing labeled data for the tasks of interest, and we would like the methods we develop to be easily applied in multiple domains. Motivated by this, we propose a generative model for solving these tasks jointly without labeled data. Interestingly, we find that jointly modeling refinements at both the review and segment level is beneficial. We created a new recipe data set, and manually labeled a random sample to evaluate our model and several baselines. We obtain 90.1% F_1 for predicting refinements at the review level, and 77.0% F_1 for predicting refinement segments within reviews.

2 Related Work

At first glance, the task of identifying refinements appears similar to subjectivity detection (see (Pang and Lee, 2008) for a survey). However, note that an objective sentence is not necessarily a refinement: e.g., "I took the cake to work"; and a subjective sentence can still contain a refinement: e.g., "I reduced the sugar and it came out perfectly."

Our end goal is similar to review summarization. However, previous work on review summarization (Hu and Liu, 2004; Popescu and Etzioni, 2005; Titov and McDonald, 2008) in product or service domains focused on summarizing evaluative information — more specifically, identifying ratable aspects (e.g., "food" and "service" for restaurants) and summarizing the overall sentiment polarity for each aspect. In contrast, we are interested in extracting a subset of the non-evaluative information. Rather than ratable aspects that are common across the entire domain (e.g., "ingredient", "cooking method"), we are interested in actionable information that is related and *specific* to the subject of the review.

Note that while our end goal is to summarize objective information, it is still very different from standard multi-document summarization (Radev et al., 2002) of news articles. Apart from differences in the quantity and the nature of the input, we aim to summarize a distribution over what should or can be changed, rather than produce a consensus using different accounts of an event. In terms of modeling approaches, in the context of extractive summarization, Barzilay and Lee (2004) model content structure (i.e., the order in which topics appear) in documents. We also model document structure, but we do so to help identify refinement segments.

We share with previous work on predicting review quality or helpfulness an interest in identifying "informative" text. Early work tried to exploit the intuition that a helpful review is one that comments on product details. However, incorporating product-aspect-mention count (Kim et al., 2006) or similarity between the review and product specification (Zhang and Varadarajan, 2006) as features did not seem to improve the performance when the task was predicting the percentage of helpfulness votes. Instead of using the helpfulness votes, Liu et al. (2007) manually annotated reviews with quality judgements, where a *best* review was defined as one that contains complete and detailed comments. Our notion of informativeness differs from previous work. We do not seek reviews that contain detailed *evaluative* information; instead, we seek reviews that contain detailed *actionable* information. Furthermore, we are not expecting any single review to be comprehensive; rather, we seek to extract a collection of refinements representing the collective wisdom of the community.

To the best of our knowledge, there is little previous work on mining user-generated data for actionable information. However, there has been increasing interest in language grounding. In particular, recent work has studied learning to act in an external environment by following textual instructions (Branavan et al., 2009, 2010, 2011; Vogel and Jurafsky, 2010). This line of research is complementary to our work. While we do not utilize extensive linguistic knowledge to analyze actionable information, we view this is an interesting future direction.

We propose a generative model that makes predictions at both the review and review segment level. Recent work uses a discriminative model with a similar structure to perform sentence-level sentiment analysis with review-level supervision (Täckström and McDonald, 2011). However, sentiment polarity labels at the review level are easily obtained. In contrast, refinement labels are not naturally available, motivating the use of unsupervised learning. Note that the model of Täckström and McDonald (2011) cannot be used in a fully unsupervised setting.

3 Refinements

In this section, we define refinements more precisely. We use recipes as our running example, but our problem formulation and models are not specific to this domain.

A *refinement* is a piece of text containing actionable information that is not entailed by the original instructions, but can be used to modify or expand the original instructions. A refinement could propose an alternative method or an improvement (e.g., "I replaced half of the shortening with butter", "Let the shrimp sit in 1/2 marinade for 3 hours"), as well as provide clarification ("definitely use THIN cut pork chops, otherwise your panko will burn before your chops are cooked").

Furthermore, we distinguish between a *verified* refinement (what the user actually did) and a *hypothetical* refinement ("next time I think I will try evaporated milk"). In domains similar to recipes, where instructions may be carried out repeatedly, there exist refinements in both forms. Since instructions should, in principle, contain information that has been well tested, in this work, we consider only the former as our target class. In a small percentage of reviews we observed "failed attempts" where a user did not follow a certain step and regretted the diversion. In this work, we do not consider them to be refinements. We refer to text that does not contain refinements as *background*.

Finally, we note that the presence of a past tense verb does not imply a refinement (e.g., "Everyone *loved* this dish", "I *got* many compliments"). In fact, not all text segments that describe an action are refinements (e.g., "I *took* the cake to work", "I *followed* the instructions to a T").

4 Models

In this section we describe our models. To identify refinements without labeled data, we propose a generative model of reviews (or more generally documents) with latent variables. We assume that each review $\mathbf{x}$ is divided into *segments*, $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_T)$. Each segment is a sub-sentence-level text span. We assume that the segmentation is observed, and hence it is not modeled. The segmentation procedure we use is described in Section 5.1.

While we focus on the unsupervised setting, note that the model can also be used in a semi-supervised setting. In particular, coarse (review-level) labels can be used to guide the induction of fine-grained latent structure (segment labels, alignments).

4.1 Identifying Refinements

We start by directly modeling refinements at the segment level. Our first intuition is that refinement and background segments can often be identified by lexical differences. Based on this intuition, we can ignore document structure and generate the segments with a segment-level mixture of multinomials (S-Mix). In general we could use n multinomials to represent refinements and m multinomials to represent background text, but in this paper we simply use $n = m = 1$. Therefore, unsupervised learning in S-Mix can be viewed as clustering the segments with two latent states. As is standard practice in unsupervised learning, we subsequently map these latent states onto the labels of interest: r and b, for refinement and background, respectively. Note, however, that this model ignores potential sequential dependencies among segments. A segment following a refinement segment in a review may be more likely to be a refinement than background, for example.

To incorporate this intuition, we could instead generate reviews with a HMM (Rabiner, 1989) over segments (S-HMM) with two latent states. Let z_i be the latent label variable for the ith segment. The joint probability of a review and segment labeling is

$$p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) = \prod_{j=1}^{T} p(z_j | z_{j-1}; \boldsymbol{\theta}) p(\mathbf{x}_j | z_j; \boldsymbol{\theta}), \quad (1)$$

where $p(z_j | z_{j-1}; \boldsymbol{\theta})$ are multinomial transition distributions, allowing the model to learn that $p(z_j = r | z_{j-1} = r; \boldsymbol{\theta}) > p(z_j = b | z_{j-1} = r; \boldsymbol{\theta})$ as motivated above, and $p(\mathbf{x}_j | z_j; \boldsymbol{\theta})$ are multinomial emission distributions. Note that all words in a segment are generated independently conditioned on z_j.

While S-HMM models sequential dependencies, note that it imposes the same transition probabilities on each review. In a manually labeled random sample of recipe reviews, we find that refinement segments tend to be clustered together in certain reviews ("bursty"), rather than uniformly distributed across all reviews. Specifically, while we estimate that 23% of all segments are refinements, 42% of reviews do not contain any refinements. In reviews that contain a refinement, 34% of segments are refinements. S-HMM cannot model this phenomenon.

Consequently, we extend S-HMM to include a latent label variable y for each review that takes values yes (*contains refinement*) and no (*does not contain refinement*). The extended model is a mixture of HMMs (RS-MixHMM) where y is the mixture component.

$$p(\mathbf{x}, y, \mathbf{z}; \boldsymbol{\theta}) = p(y; \boldsymbol{\theta}) p(\mathbf{x}, \mathbf{z} | y; \boldsymbol{\theta}) \quad (2)$$

The two HMMs $p(\mathbf{x}, \mathbf{z} \mid y = yes; \boldsymbol{\theta})$ and $p(\mathbf{x}, \mathbf{z} \mid y = no; \boldsymbol{\theta})$ can learn different transition multinomials and consequently different distributions over $\mathbf{z}$ for different y. On the other hand, we do not believe the textual content of the background segments in a $y = yes$ review should be different from those in a $y = no$ review. Thus, the emission distributions are shared between the two HMMs, $p(\mathbf{x}_j | z_j, y; \boldsymbol{\theta}) = p(\mathbf{x}_j | z_j; \boldsymbol{\theta})$.

Note that the definition of y imposes additional constraints on RS-MixHMM: 1) reviews with $y = no$ cannot contain refinement segments, and 2) reviews with $y = yes$ must contain at least one refinement segment. We enforce constraint (1) by disallowing refinement segments $z_j = r$ when $y = no$: $p(z_j = r | z_{j-1}, y = no; \boldsymbol{\theta}) = 0$. Therefore, with one background label, only the all background label sequence has non-zero probability when $y = no$. Enforcing constraint (2) is more challenging, as the $y = yes$ HMM must assign zero probability when all segments are background, but permit background segments when refinement segments are present.

To enforce constraint (2), we "rewire" the HMM structure for $y = yes$ so that a path that does not go through the refinement state r is impossible. We first expand the state representation by replacing b with two states that encode whether or not the first r has been encountered yet: $b_{not-yet}$ encodes that all previous states in the path have also been background; b_{ok} encodes that at least one refinement state has been encountered[2]. We prohibit paths from ending with $b_{not-yet}$ by augmenting RS-MixHMM with a special final state f, and fixing $p(z_{T+1} = f | z_T = b_{not-yet}, y = yes; \boldsymbol{\theta}) = 0$. Furthermore, to enforce the correct semantics of each state, paths cannot start with b_{ok}, $p(z_1 = b_{ok} | y = yes; \boldsymbol{\theta}) = 0$, and transitions from $b_{not-yet}$ to b_{ok}, b_{ok} to $b_{not-yet}$, and r to $b_{not-yet}$ are prohibited.

Note that RS-MixHMM also generalizes to the case where there are multiple refinement ($n > 1$) and background ($m > 1$) labels. Let $\mathcal{Z}_r$ be the set of refinement labels, and $\mathcal{Z}_b$ be the set of background labels. The transition structure is analogous to the $n = m = 1$ case, but statements involving r are applied for each $z \in \mathcal{Z}_r$, and statements involving b are applied for each $z \in \mathcal{Z}_b$. For example, the $y = yes$ HMM contains $2|\mathcal{Z}_b|$ background states.

In summary, the generative process of RS-MixHMM involves first selecting whether the review will contain a refinement. If the answer is yes, a sequence of background segments and at least one refinement segment are generated using the $y = yes$ HMM. If the answer is *no*, only background segments are generated. Interestingly, by enforcing constraints (1) and (2), we break the label symmetry that necessitates mapping latent states onto labels

[2] In this paper, the two background states share emission multinomials, $p(\mathbf{x}_j | z_j = b_{not-yet}; \boldsymbol{\theta}) = p(\mathbf{x}_j | z_j = b_{ok}; \boldsymbol{\theta})$, though this is not required.

when using S-Mix and S-HMM. Indeed, in the experiments we present in Section 5.3, mapping is not necessary for RS-MixHMM.

Note that the relationship between document-level labels and segment-level labels that we model is related to the *multiple-instance* setting (Dietterich et al., 1997) in the machine learning literature. In multiple-instance learning (MIL), rather than having explicit labels at the instance (e.g., segment) level, labels are given for bags of instances (e.g., documents). In the binary case, a bag is *negative* only if all of its instances are *negative*. While we share this problem formulation, work on MIL has mostly focussed on supervised learning settings, and thus it is not directly applicable to our unsupervised setting. Foulds and Smyth (2011) propose a generative model for MIL in which the generation of the bag label y is conditioned on the instance labels $\mathbf{z}$. As a result of this setup, their model reduces to our S-Mix baseline in a fully unsupervised setting.

Finally, although we motivated including the review-level latent variable y as a way to improve segment-level prediction of $\mathbf{z}$, note that predictions of y are useful in and of themselves. They provide some notion of review *usefulness* and can be used to filter reviews for search and browsing. They additionally give us a way to measure whether a set of instructions is often modified or performed as specified. Finally, if we want to provide supervision, it is much easier to annotate whether a review contains a refinement than to annotate each segment.

4.2 Alignment with the Instructions

In addition to the review $\mathbf{x}$, we also observe the set of instructions $\mathbf{s}$ being discussed. Often a review will reference specific parts of the instructions. We assume that each set of instructions is segmented into *steps*, $\mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_S)$. We augment our model with latent *alignment* variables $\mathbf{a} = (a_1, \ldots, a_T)$, where $a_j = \ell$ denotes that the jth review segment is referring to the ℓth step of $\mathbf{s}$. We also define a special NULL instruction step. An alignment to NULL signifies that the segment does not refer to a specific instruction step. Note that this encoding assumes that each review segment refers to at most one instruction step. Alignment predictions could facilitate further analysis of how refinements affect the instructions, as well as aid in summarization and visualization of refinements.

The joint probability under the augmented model, which we refer to as RSA-MixHMM, is

$$p(\mathbf{a}, \mathbf{x}, y, \mathbf{z}|\mathbf{s}; \boldsymbol{\theta}) = p(y; \boldsymbol{\theta}) p(\mathbf{a}, \mathbf{x}, \mathbf{z}|y, \mathbf{s}; \boldsymbol{\theta}) \qquad (3)$$

$$p(\mathbf{a}, \mathbf{x}, \mathbf{z}|y, \mathbf{s}; \boldsymbol{\theta}) = \prod_{j=1}^{T} p(a_j, z_j|a_{j\text{–}1}, z_{j-1}, y, \mathbf{s}; \boldsymbol{\theta}) \times p(\mathbf{x}_j|a_j, z_j, \mathbf{s}; \boldsymbol{\theta}).$$

Note that the instructions $\mathbf{s}$ are assumed to be observed and hence are not generated by the model. RSA-MixHMM can be viewed as a mixture of HMMs where each state encodes both a segment label z_j and an alignment variable a_j. Encoding an alignment problem as a sequence labeling problem was first proposed by Vogel et al. (1996). Note that RSA-MixHMM uses a similar expanded state representation and transition structure as RS-MixHMM to encode the semantics of y.

In our current model, the transition probability decomposes into the product of independent label transition and alignment transition probabilities

$$p(a_j, z_j|a_{j-1}, z_{j-1}, y, \mathbf{s}; \boldsymbol{\theta}) = p(a_j|a_{j-1}, y, \mathbf{s}; \boldsymbol{\theta}) \times p(z_j|z_{j-1}, y, \mathbf{s}; \boldsymbol{\theta}),$$

and $p(a_j|a_{j-1}, y, \mathbf{s}; \boldsymbol{\theta}) = p(a_j|y, \mathbf{s}; \boldsymbol{\theta})$ simply encodes the probability that segments align to a (non-NULL) instruction step given y. This allows the model to learn, for example, that reviews that contain refinements refer to the instructions more often.

Intuitively, a segment and the step it refers to should be lexically similar. Consequently, RSA-MixHMM generates segments using a mixture of the multinomial distribution for the segment label z_j and the (fixed) multinomial distribution[3] for the step $\mathbf{s}_{a_j}$. In this paper, we do not model the mixture probability and simply assume that all overlapping words are generated by the instruction step. When $a_j =$ NULL, only the segment label multinomial is used. Finally, we disallow an alignment to a non-NULL step if no words overlap: $p(\mathbf{x}_j|a_j, z_j, \mathbf{s}; \boldsymbol{\theta}) = 0$.

4.3 Inference and Parameter Estimation

Because our model is tree-structured, we can efficiently compute exact marginal distributions

[3] Stopwords are removed from the instruction step.

over latent variables using the *sum-product algorithm* (Koller and Friedman, 2009). Similarly, to find maximum probability assignments, we use the *max-product algorithm*.

At training time we observe a set of reviews and corresponding instructions, $\mathcal{D} = \{(\mathbf{x}^1, \mathbf{s}^1), \ldots, (\mathbf{x}^N, \mathbf{s}^N)\}$. The other variables, y, $\mathbf{z}$, and $\mathbf{a}$, are latent. For all models, we estimate parameters to maximize the marginal likelihood of the observed reviews. For example, for RSA-MixHMM, we estimate parameters using

$$\arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log \sum_{\mathbf{a},\mathbf{z},y} p(\mathbf{a}, \mathbf{x}^i, y, \mathbf{z} | \mathbf{s}^i; \boldsymbol{\theta}).$$

This problem cannot be solved analytically, so we use the Expectation Maximization (EM) algorithm.

5 Experiments

5.1 Data

In this paper, we use recipes and reviews from *allrecipes.com*, an active community where we estimate that the mean number of reviews per recipe is 54.2. We randomly selected 22,437 reviews for our data set. Of these, we randomly selected a subset of 550 reviews and determined whether or not each contains a refinement, using the definition provided in Section 3. In total, 318 of the 550 (57.8%) contain a refinement. We then randomly selected 119 of the 550 and labeled the individual segments. Of the 712 segments in the selected reviews, 165 (23.2%) are refinements and 547 are background.

We now define our review segmentation scheme. Most prior work on modeling latent document substructure uses sentence-level labels (Barzilay and Lee, 2004; Täckström and McDonald, 2011). In the recipe data, we find that sentences often contain both refinement and background segments: "[*I used a slow cooker with this recipe and*] [it turned out great!]" Additionally, we find that sentences often contain several distinct refinements: "[*I set them on top and around the pork and*] [*tossed in a can of undrained french cut green beans and*] [*cooked everything on high for about 3 hours*]." To make refinements easier to identify, and to facilitate downstream processing, we allow sub-sentence segments.

Our segmentation procedure leverages a phrase structure parser. In this paper we use the Stanford Parser[4]. Based on a quick manual inspection, domain shift and ungrammatical sentences do cause a significant degradation in parsing accuracy when compared to in-domain data. However, this is acceptable because we only use the parser for segmentation. We first parse the entire review, and subsequently iterate through the tokens, adding a segment break when any of the following conditions is met:

- sentence break (determined by the parser)
- token is a *coordinating conjunction* (CC) with parent other than NP, PP, ADJP
- token is a *comma* (,) with parent other than NP, PP, ADJP
- token is a *colon* (:)

The resulting segmentations are fixed during learning. In future work we could extend our model to additionally identify segment boundaries.

5.2 Experimental Setup

We first describe the methods we evaluate. For comparison, we provide results with a baseline that randomly guesses according to the class distribution for each task. We also evaluate a **R**eview-level model:

- **R-Mix**: A review-level mixture of multinomials with two latent states.

Note that this is similar to clustering at the review level, except that class priors are estimated. R-Mix does not provide segment labels, though they can be obtained by labeling all segments with the review label.

We also evaluate the two **S**egment-level models described in Section 4.1 (with two latent states):

- **S-Mix**: A segment-level mixture model.
- **S-HMM**: A segment-level HMM (Eq. 1).

These models do not provide review labels. To obtain them, we assign $y = yes$ if any segment is labeled as a refinement, and $y = no$ otherwise.

Finally, we evaluate three versions of our model (**R**eview + **S**egment and **R**eview + **S**egment +

[4] http://nlp.stanford.edu/software/lex-parser.shtml

Alignment) with one refinement segment label and one background segment label[5]:

- **RS-MixHMM**: A mixture of HMMs (Eq. 2) with constraints (1) and (2) (see Section 4).
- **RS-MixMix**: A variant of RS-MixHMM without sequential dependencies.
- **RSA-MixHMM**: The full model that also incorporates alignment (Eq. 3).

Segment multinomials are initialized with a small amount of random noise to break the initial symmetry. RSA-MixHMM segment multinomials are instead initialized to the RS-MixHMM solution. We apply add-0.01 smoothing to the emission multinomials and add-1 smoothing to the transition multinomials in the M-step. We estimate parameters with 21,887 unlabeled reviews by running EM until the relative percentage decrease in the marginal likelihood is $\leq 10^{-4}$ (typically 10-20 iterations).

The models are evaluated on refinement F_1 and accuracy for both review and segment predictions using the annotated data described in Section 5.1. For R-Mix and the segment (*S*-) models, we select the 1:1 mapping of latent states to labels that maximizes F_1. For RSA-MixHMM and the RS- models this was not necessary (see Section 4.1).

5.3 Results

Table 1 displays the results. R-Mix fails to accurately distinguish refinement and background reviews. The words that best discriminate the two discovered review classes are "savory ingredients" (*chicken, pepper, meat, garlic, soup*) and "baking/dessert ingredients" (*chocolate, cake, pie, these, flour*). In other words, reviews naturally cluster by topics rather than whether they contain refinements.

The segment models (S-) substantially outperform R-Mix on all metrics, demonstrating the benefit of segment-level modeling and our segmentation scheme. However, S-HMM fails to model the "burstiness" of refinement segments (see Section 4.1). It predicts that 76.2% of reviews contain refinements, and additionally that 40.9% of segments contain refinements, whereas the true values are 57.8% and 23.2%, respectively. As a result, these models provide high recall but low precision.

In comparison, our models, which model the review labels[6] y, yield more accurate refinement predictions. They provide statistically significant improvements in review and segment F_1, as well as accuracy, over the baseline models. RS-MixHMM predicts that 62.9% of reviews contain refinements and 28.2% of segments contain refinements, values that are much closer to the ground truth. The refinement emission distributions for S-HMM and RS-MixHMM are fairly similar, but the probabilities of several key terms like *added*, *used*, and *instead* are higher with RS-MixHMM.

The review F_1 results demonstrate that our models are able to very accurately distinguish refinement reviews from background reviews. As motivated in Section 4.1, there are several applications that can benefit from review-level predictions directly. Additionally, note that review labeling is not a trivial task. We trained a supervised logistic regression model with bag-of-words and length features (for both the number of segments and the number of words) using 10-fold cross validation on the labeled dataset. This supervised model yields mean review F_1 of 78.4, 11.7 F_1 points below the best unsupervised result[7].

Augmenting RS-MixMix with sequential dependencies, yielding RS-MixHMM, provides a moderate (though not statistically significant) improvement in segment F_1. RS-MixHMM learns that refinement reviews typically begin and end with background segments, and that refinement segments tend to appear in succession.

RSA-MixHMM additionally learns that segments in refinement reviews are more likely to align to non-NULL recipe steps. It also encourages the segment multinomials to focus modeling effort on words that appear only in the reviews. As a result, in addition to yielding alignments, RSA-MixHMM provides small improvements over RS-MixHMM (though they are not statistically significant).

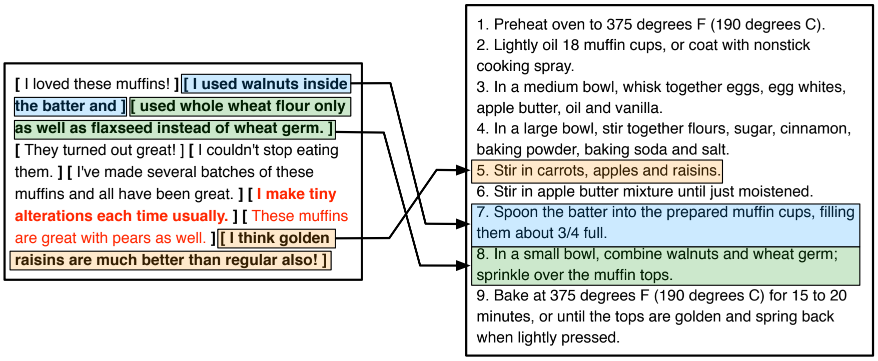
[5] Attempts at modeling refinement and background subtypes by increasing the number of latent states failed to substantially improve the results.

[6] We note that enforcing the constraint that a refinement review must contain at least one refinement segment using the method in Section 4.1 provides a statistically significant significant improvement in review F_1 of 4.0 for RS-MixHMM.

[7] Note that we do not consider this performance to be the upper-bound of supervised approaches; clearly, supervised approaches could benefit from additional labeled data. However, labeled data is relatively expensive to obtain for this task.

Model	**review** (57.8% refinement)				**segment** (23.2% refinement)			
	acc	**prec**	**rec**	**F$_1$**	**acc**	**prec**	**rec**	**F$_1$**
random baseline	51.2†	57.8	57.8	57.8†	64.4†	23.2	23.2	23.2†
R-Mix	61.5†	69.1	60.4	64.4†	55.8†	27.9	57.6	37.6†
S-Mix	77.5†	72.4	**98.7**	83.5†	80.6†	54.7	95.2	69.5†
S-HMM	79.8†	74.7	98.4	84.9†	80.3†	54.3	**95.8**	69.3†
RS-MixMix	87.1	85.4	93.7	89.4	86.4	65.6	86.7	74.7
RS-MixHMM	87.3	85.6	93.7	89.5	87.9	69.7	84.8	76.5
RSA-MixHMM	**88.2**	**87.1**	93.4	**90.1**	**88.5**	**71.7**	83.0	**77.0**

Table 1: Unsupervised experiments comparing models for review and segment refinement identification on the recipe data. Bold indicates the best result, and a † next to an accuracy or F_1 value indicates that the improvements obtained by RS-MixMix, RS-MixHMM, and RSA-MixHMM are significant ($p = 0.05$ according to a bootstrap test).



[I loved these muffins!] **[I used walnuts inside the batter and]** **[used whole wheat flour only as well as flaxseed instead of wheat germ.]** [They turned out great!] [I couldn't stop eating them.] [I've made several batches of these muffins and all have been great.] **[I make tiny alterations each time usually.]** [These muffins are great with pears as well.] **[I think golden raisins are much better than regular also!]**

1. Preheat oven to 375 degrees F (190 degrees C).
2. Lightly oil 18 muffin cups, or coat with nonstick cooking spray.
3. In a medium bowl, whisk together eggs, egg whites, apple butter, oil and vanilla.
4. In a large bowl, stir together flours, sugar, cinnamon, baking powder, baking soda and salt.
5. Stir in carrots, apples and raisins.
6. Stir in apple butter mixture until just moistened.
7. Spoon the batter into the prepared muffin cups, filling them about 3/4 full.
8. In a small bowl, combine walnuts and wheat germ; sprinkle over the muffin tops.
9. Bake at 375 degrees F (190 degrees C) for 15 to 20 minutes, or until the tops are golden and spring back when lightly pressed.

Figure 1: Example output (best viewed in color). Bold segments in the review (left) are those predicted to be refinements. Red indicates an incorrect segment label, according to our gold labels. Alignments to recipe steps (right) are indicated with colors and arrows. Segments without colors and arrows align to the NULL recipe step (see Section 4.2).

We provide an example alignment in Figure 1. Annotating ground truth alignments is challenging and time-consuming due to ambiguity, and we feel that the alignments are best evaluated via a downstream task. Therefore, we leave thorough evaluation of the quality of the alignments to future work.

6 Conclusion and Future Work

In this paper, we developed unsupervised methods based on generative models for mining refinements to online instructions from reviews. The proposed models leverage lexical differences in refinement and background segments. By augmenting the base models with additional structure (review labels, alignments), we obtained more accurate predictions.

However, to further improve accuracy, more linguistic knowledge and structure will need to be incorporated. The current models provide many false positives in the more subtle cases, when some words that typically indicate a refinement are present, but the text does not describe a refinement according to the definition in Section 3. Examples include hypothetical refinements ("next time I will substitute...") and discussion of the recipe without modification ("I found it strange to... but it worked ...", "I love balsamic vinegar and herbs", "they baked up nicely").

Other future directions include improving the alignment model, for example by allowing words in the instruction step to be "translated" into words in the review segment. Though we focussed on recipes, the models we proposed are general, and could be applied to other domains. We also plan to consider this task in other settings such as online forums, and develop methods for summarizing refinements.

Acknowledgments

We thank Andrei Broder and the anonymous reviewers for helpful discussions and comments.

References

Regina Barzilay and Lillian Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Proceedings of the Main Conference*, pages 113–120, 2004.

S.R.K Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2009.

S.R.K Branavan, Luke Zettlemoyer, and Regina Barzilay. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2010.

S.R.K. Branavan, David Silver, and Regina Barzilay. Learning to win by reading manuals in a monte-carlo framework. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2011.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1 - 2):31 – 71, 1997.

J. R. Foulds and P. Smyth. Multi-instance mixture models and semi-supervised learning. In *SIAM International Conference on Data Mining*, 2011.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 168–177, 2004.

Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 423–430, 2006.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. Low-quality product review detection in opinion summarization. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 334–342, 2007.

Bo Pang and Ravi Kumar. Search in the lost sense of query: Question formulation in web search queries and its temporal changes. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2011.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, 2005.

Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Dragomir R. Radev, Eduard Hovy, and Kathleen McKeown. Introduction to the special issue on summarization. *Computational Linguistics*, 28 (4):399–408, 2002. ISSN 0891-2017.

Oscar Täckström and Ryan McDonald. Discovering fine-grained sentiment with latent variable structured prediction models. In *Proceedings of the 33rd European conference on Advances in information retrieval*, ECIR'11, pages 368–374, 2011.

Ivan Titov and Ryan McDonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2008.

Adam Vogel and Daniel Jurafsky. Learning to follow navigational directions. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2010.

Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, pages 836–841, 1996.

Zhu Zhang and Balaji Varadarajan. Utility scoring of product reviews. In *Proceedings of the ACM SIGIR Conference on Information and Knowledge Management (CIKM)*, pages 51–57, 2006.