The AAAI-17 Workshop on
Human-Aware Artificial Intelligence
WS-17-10

Learning to Suggest Phrases

Kenneth C. Arnold
Harvard CS
Cambridge, MA

Kai-Wei Chang
University of Virginia
Charlottesville, VA

Adam T. Kalai
Microsoft Research
Cambridge, MA

Abstract

Intelligent keyboards can support writing by suggesting content. Certain types of phrases, when offered as suggestions, may be systematically chosen more often than their frequency in a corpus of text would predict. In order to generate those types of suggestions, we collected a dataset of how human authors responded to suggestions offered to them during open-ended writing tasks. We present an offline strategy for evaluating suggestions that enables us to learn the parameters of an improved suggestion generation policy without the expense of collecting additional data under that policy. We validate the approach by simulation and on human data by demonstrating improvement in held-out suggestion acceptance rate. Our approach can be applied to other scenarios where what is typical is not necessarily what is desirable.

Introduction

Intelligent systems can help us write by proactively suggesting words or phrases while we type. For example, the virtual keyboards widely deployed on contemporary touchscreen devices almost always offer suggestions of the next word (or completions of partial words) that can be accepted with a single tap. Although word suggestions actually slow down typing on average (because of the cost of attending to the suggestions), they save effort and users prefer them (Quinn and Zhai 2016). Recent work has shown that by augmenting the user interface as illustrated in Figure 1, writers often take advantage of phrase suggestions as long as five or six words (Arnold, Gajos, and Kalai 2016). Interestingly, with longer phrases, writers accept suggestions even when they were not exactly what they intended to type themselves (Arnold, Gajos, and Kalai 2016). For example, some participants reported that the suggestions have ideas for new topics or creative wordings. But not all suggestions were good or helpful: participants found some suggested phrases boring and trite—primarily because the suggestions were generated by picking the *most likely* phrases according to a traditional language model trained to predict the next word based on previous words in context. Ironically, the predictions may have been *too accurate*; for helping people create, showing the most likely next step may be uninspiring.

Figure 1: We address a new task in human-centered AI: how to offer *suggestions* during text composition. The above example illustrates how effective suggestions may differ from accurate predictions. Even though the middle suggestion is predicted (by an n-gram model) to be about 1,000 times more likely than the one on the right, the author in our experiment chose to triple-tap on the right button, thereby inserting the words "nice change from". (This interface augments the standard mobile phone suggestion bar by providing a preview for the coming words shown below in small blue text.)

How can a system generate *good* suggestions that are likely to be accepted by writers? We propose this challenge as a new task for AI research. In this paper we address two primary obstacles facing this task: how to *obtain data* about how authors respond to suggestions, and how to *learn models* to identify and generate good suggestions from this data.

The obvious approach to obtain data would be to use existing text corpora. However, since suggestions can affect even the topics that authors discuss, existing text created without suggestions is of limited use in evaluating whether a suggestion is good. In particular, accurate predictions may be poor suggestions. For example, at the beginning of a sentence of a restaurant review,"I love this place" is a reasonable prediction, but a review writer might prefer a suggestion of a much less likely phrase such as "This was truly a wonderful experience"—they may simply not have thought of

this more enthusiastic phrase or taken the additional effort to type it. (Both examples were drawn from our dataset.) One alternative would be to show human annotators suggestions in context and ask them to decide which suggestions are most likely to be accepted. A drawback of this approach is that behavior while using a system may differ greatly from offline assessments of how one imagines one (or others) would have behaved.

To obtain data about how authors respond to suggestions, we asked crowd workers to write reviews of a specific well-known restaurant. While they were writing those reviews, our interface (Figure 1) suggests phrases generated by a language model trained on restaurant reviews from the Yelp Academic Dataset.[1] We logged which phrases were suggested along with the context, the resulting reward (words accepted), and their probability of generation according to the policy used when collecting the data (the "reference policy"). We release this data for other researchers to use.[2]

But even with this data, it is not trivial to learn to generate good suggestions, since the data was collected while offering suggestions according to a different policy than the one being learned. Traditionally we would have had to test any proposed new policy in an online experiment such as an A/B test, but that requires repeated expensive trials. Instead, we build on insights from multi-world testing (MWT) in contextual bandit learning (Li et al. 2011), which uses logs of actions taken under one stochastic "reference policy" to compute an unbiased estimate (using importance weighting) of how any other policy would perform in an A/B test without having to run that test. Hence, model parameters can be fit once using a single dataset, rather than having to run repeated A/B tests to quantify each parameter improvement. We adapt MWT to learn to generate suggestions: intuitively, for every suggestion that gets accepted, the model tries to learn to increase the likelihood that it would have generated that suggestion, weighted by the *inverse propensity* that the reference policy generated that suggestion. But since there are exponentially many possible phrase suggestions in our setting, the variance of the inverse propensity estimator can be very large. To bound the estimator's variance, we truncate our importance weights, as has been done in related contexts (Ionides 2008; Swaminathan and Joachims 2015). We then use insights from discriminative language modeling (Roark et al. 2004) to apply this approach to generating language.

Our contributions are summarized below:

- We present a new AI application of *phrase suggestion* for specific writing domains on mobile phones. We release the dataset to facilitate further research on this topic.
- We present TIP (Truncated Inverse Propensity sampling), an offline method for goal-directed training of generative language models.
- We propose a simple log-linear language model that can incorporate additional word-level features trained to offer suggestions that will be accepted, rather than to predict, and show that it increases acceptance rate over various baselines both in simulation and on our dataset.

[1] https://www.yelp.com/dataset_challenge

[2] http://tiny.cc/arnold17suggestions-data

Related work

Generative language models have a long history (Reiter, Dale, and Feng 2000) and play an important role in many NLP applications, including response generation in social networks (Sordoni et al. 2015), dialog systems (Rambow, Bangalore, and Walker 2001), summarization (Mani 2001), and translation (Johnson et al. 2016). Prior work usually treats humans as a source of training or validation data; we are more interested in designing *interactive* systems that people find helpful. We do not directly study whether people find our system helpful either, but we optimize a proxy measure: how many of the system's suggestions they accept.

Based on the theory that people want to see accurate predictions of the current or upcoming word, deployed keyboards use n-gram language models (Quinn and Zhai 2016; Kneser and Ney 1995), or sometimes neural language models (Kim et al. 2016), trained to predict the next word given recent context. Recent advances in language modeling have increased the accuracy of these predictions by, e.g., using additional context (Mikolov and Zweig 2012). However, these approaches all rely on the assumption that accurate predictions are most helpful suggestions, which (Arnold, Gajos, and Kalai 2016) calls into question.

The difference between suggestion and prediction is more pronounced when showing phrases rather than just words. Prior work has extended predictive language modeling to phrase prediction (Nandi and Jagadish 2007) and sentence completion (Bickel, Haider, and Scheffer 2005), but to our knowledge did not evaluate whether people found those systems helpful. Google's "Smart Reply" email response suggestion system (Kannan et al. 2016) avoids showing a likely predicted response if it is too similar to one of the options already presented, but the approach is heuristic, based on a priori similarity. Search engine query completion also generates long phrases that can function as suggestions, but is typically trained to predict what query is eventually made (e.g., (Jiang et al. 2014)). Writing assistant systems also try to generate or retrieve helpful text based on heuristics (Liu et al. 2000; Chen et al. 2012).

Language generation systems have not tried to optimize their behavior for human acceptance because this previously required expensive A/B testing. However, Li et al. showed how to learn a news article recommendation policy based on the limited feedback of users interacting with a small number of articles with an appropriate randomized logging scheme and inverse propensity sampling (2011).

Additionally, we take a log-linear language model, previously used to increase predictive accuracy in discriminative language modeling by leveraging additional features (Roark et al. 2004), and adapt it to generating suggestions.

Offline Learning of Suggestion Policies

We consider systems that offer suggestions to users performing a sequential task such as writing, translating, or paraphrasing, according to a stochastic policy. This setting gen-

eralizes the setting of (Arnold, Gajos, and Kalai 2016); we will consider this setting as a running example throughout this paper.

In this section, we provide a formal definition of the phase suggestion problem and discuss an offline approach to policy evaluation and learning.

Data and notation. We denote by $|A|$ the size of set A and $\Delta(A)$ the set of probability distributions over A. A phrase suggestion *dataset* $\mathcal{D}$ consists of four-tuples (c^t, s^t, q^t, r^t), for $t = 1, 2, \ldots, T$, where $c^t \in C$ is a context, s^t is a suggestion drawn from the distribution generated with probability q^t by a stochastic *phrase suggestion policy* which we refer to as q given c^t (i.e., $s^t \sim q_{|c^t}$) and the observed reward r^t. We assume there is a fixed distribution $\mu \in \Delta(C)$ over a fixed set of contexts $c \in C$. For instance, a context can include the previously typed words, characters of the word being typed thus far, and possibly other metadata. Let S to be the space of all possible suggestions, a *conditional suggestion distribution* $p : C \to \Delta(S)$ is a distribution over suggestions for each context.[3] This distribution reflects the distribution of suggestions $s \in S$ given by a randomized suggestion algorithm whose input is context c. The distribution q^t is an instance of p, and we use it to generate suggestions s^t in the t-th example. For any suggestion s and context c, there is a *reward* $r \in [0, B]$, for some positive *reward bound* $B > 0$, that is drawn from the corresponding distribution $\rho_{|sc} \in \Delta([0, B])$. The rewards r can be task-specific, e.g., the speed that a sentence was written, or an annotator's rating of quality; in our case, we take r to be the number of words accepted.[4]

Expected Reward. Of particular interest is the *expected reward* function $R : S \times C \to [0, B]$, which is simply the expected reward for a given suggestion in a given context, $R(s, c) = \mathrm{E}_{r\sim\rho_{|sc}}[r]$. The expected reward of the conditional distribution p generated by a suggestion algorithm, which we also refer to as its *value*, is defined in terms of the underlying distribution on contexts μ, and the expected reward function R. We denote the value by $v(p; \mu, R) = \mathrm{E}_{c\sim\mu, s\sim p_{|c}}[R(s, c)]$.[5] The ultimate goal is to learn a suggestion system in which $v(p)$ is (potentially) maximal.

Offline Evaluation. The key challenge here is the reward $R(s, c)$ is only accessible through interacting with authors in actual writing tasks, and S is far too large to try every possibility. Therefore, to estimate $v(p)$, we would have to deploy a system generating suggestions from p and collect user logs. Training a system in such a manner would be impractical.

Inverse Propensity (IP) sampling methods, i.e., importance weighting, provide us a way to use a pre-obtained dataset like $\mathcal{D}$ to estimate, in an unbiased fashion, the expected reward of p, $v(p)$. Suppose we have a dataset generated by a "reference" suggestion policy q. Then we can evaluate the expected reward of a candidate policy p as:

$$\mathop{\mathrm{E}}_{\substack{c\sim\mu \\ s\sim p_{|c}}}[R(s, c)] = \mathop{\mathrm{E}}_{\substack{c\sim\mu \\ s\sim q_{|c}}}\left[R(s, c)\frac{p(s|c)}{q(s|c)}\right] \quad (1)$$

In particular, the expectation of $r^t p(s^t|c^t)/q^t$ is exactly $v(p)$. However, the variance of this estimate can be large if $p(s|c)/q(s|c)$ is large, so a prohibitively large number of training samples would be required for obtaining a good model. To reduce the variance of this estimator, we introduce *Truncated Inverse Propensity* (TIP), defined as

$$v_M(p, q, \mu, R) = \mathop{\mathrm{E}}_{\substack{c\sim\mu \\ s\sim q_{|c}}}\left[R(s, c)\min\left(\frac{p(s|c)}{q(s|c)}, M\right)\right], \quad (2)$$

written $v_M(p)$ when μ, R, and q are clear from context. Note that $v_M(p) \le v(p)$ so v_M is a lower-bound on the value of a suggestion algorithm. Moreover, this truncated estimate $v_M(p)$ can be well-approximated using a fixed dataset with a bounded number of samples. We will show the theoretical properties and empirical performance of learning with TIP in the remainder of the paper. Eq. (2) suggests that we can evaluate $v(p)$ based on a dataset $\mathcal{D} = (c^t, s^t, q^t, r^t)_{t=1}^T$ generated by reference distribution q:

$$\tilde{v}_M(p) = \frac{1}{T}\sum_{t=1}^{T}\left[r^t \min\left(\frac{p(s^t|c^t)}{q^t(s^t|c^t)}, M\right)\right].$$

When T is sufficiently large, $\tilde{v}_M(p)$ approximates $v_M(p)$. Moreover, as T increases, we can afford to have a larger value of M while maintaining accurate estimates of $v_M(p)$, which approaches $v(p)$ as M increases without bound.

Assumptions. We make several simplifying assumptions and leave these directions to future work. First, we assume that the distribution over contexts is fixed and not influenced by prior suggestions. How past suggestions will influence future writing is a fascinating issue — once a suggestion interface begins to dramatically influence the distribution over content generated, it has already achieved our motivating goal of having suggestions accepted. Second, practical interfaces such as the one we use in our experiments may offer more than one suggestion at a time; the formulation above assumes that suggestions do not interact[6]. Some related issues such as diversity of suggestions and building personalized system have been studied in information retrieval and machine learning (e.g., (Shokouhi 2013; Guzman-Rivera et al. 2014)).

[3] We write $p_{|c}(s)$ or $p(s|c)$ interchangeably.

[4] Accepting a word after typing a prefix of it gets a reward of the fraction of words in the vocabulary that start with that prefix.

[5] We write $v(p) = v(p; \mu, R)$ when μ and R are understood from context.

[6] We experimented with a variation of this mathematical setting that accounted for the group of three suggestions that our interface actually offered, but it was unclear how the generation probabilities of the other two suggestions should be updated.

Learning with Truncated Inverse Propensity. In the following, we show the theoretical properties of learning with TIP. Suppose the suggestion policy p is parameterized by $\theta \in \Theta$. During the learning, we aim to maximize the *Truncated Inverse Propensity* (TIP), defined as $\hat{r}^t_\theta = r^t \min\left(p_\theta(s|c)/q^t, M\right)$ for the t-th example.

Without truncating the importance weights, the model could easily overfit. For example, if the model could isolate and assign probability 1 to a single accepted suggestion, its reward could be $1/q$, which could be very large if q is small. Truncating avoids overfitting by a kind of regularization in the space of suggestion distributions. Following the ideas in (Ionides 2008; Swaminathan and Joachims 2015), the following lemma shows that maximizing $v_M(p) \approx \sum_t \hat{r}^t/T$ is unlikely to overfit data and that it is likely to do almost as well as those p that are "nearby" q in the sense that they are rarely larger than M times q.

Lemma 1. *Let C, S be finite sets of contexts and suggestions, μ be an arbitrary distribution over contexts C, $q : C \to \Delta(S)$, $M > 0$, and $R : S \times C \to [0, B]$. Then,*

1. *For any $p : C \to \Delta(S)$, we have $v_M(p, q, \mu, R) \leq v(p, q, \mu, R)$ with equality if $p(s|c) \leq Mq(s|c)$ for all c, s.*
2. *Let $\theta^* = \arg\max_\theta v_M(p_\theta)$. Then,*

$$v(p_{\theta^*}) \geq \max_\theta v(p_\theta) - B \Pr_{\substack{c\sim\mu \\ s\sim p_{|c}}}\left[\frac{p_\theta(c|s)}{q(c|s)} > M\right]$$

3. *Let dataset $(c^1, s^1, q^1, r^1) \ldots (c^T, s^T, q^T, r^T)$ and $\hat{r}^t_\theta = r^t \min\left(p_\theta(s|c)/q^t, M\right)$. Then for any $\delta > 0$, with probability $\geq 1 - \delta$ over the choice of dataset, simultaneously for all $\theta \in \Theta$,*

$$\left| v_M(p_\theta) - \frac{1}{T}\sum_{t=1}^{T} \hat{r}^t_\theta \right| \leq BM\sqrt{\frac{\log\left(2|\Theta|/\delta\right)}{2T}}.$$

Proof. Part 1 follows by the definition of v_M and the fact that $\min(x, y) \leq x$ with equality if $x \leq y$. For part 2, note that because the reward is at most B, for any p,

$$v(p) - B \Pr_{\substack{c\sim\mu \\ s\sim p_{|c}}}\left[\frac{p(s|c)}{q(s|c)} \geq M\right] \leq v_M(p)$$

Combining with $v_M(p) \leq v_M(p_{\theta^*}) \leq v(p_{\theta^*})$ and maximizing over p implies part 2.

For part 3, note that our estimates are independent unbiased estimates of the M-bounded value, i.e., $\mathrm{E}[\hat{r}^t_\theta] = v_M(p_\theta)$. Combining this with Hoeffding's inequality and the union bound over the $|\Theta|$ different parameter values completes the proof. □

This lemma justifies maximizing $\sum_t \hat{r}^t_\theta$ over $\theta \in \Theta$. In particular, the generalization bound[7] in part 3 implies that, as long as M is not too large, we will not overfit as the M-bounded value of each parameter setting will be close to its estimate from training. Combined with part 2, it implies that the maximizing this quantity should perform at least as well as maximizing the expected reward $v(p_\theta)$ over p_θ "near" q in the sense that $p_\theta \leq Mq$. However, it is still meaningful when p_θ is sometimes greater than Mq.

[7]For simplicity, the generalization bound in 3 is stated in terms of a finite parameter set Θ (which is true when relatively few bits per floating-point parameter suffice), but could also be extended to infinite sets as long as there are bounds on complexity of class $\mathcal{P}_\Theta$.

Comparing Policies using TIP

When using TIP to evaluate performance (e.g., on held-out data), it is important to use the same threshold M and test set size for all algorithms to ensure that the comparison is unbiased. Most importantly, the choice of M is limited by the width of the resulting confidence intervals—larger values of M result in larger confidence intervals. Lemma 1 suggests that the confidence intervals grow roughly $O(M/\sqrt{T})$. In our experiments, we choose M so as to yield reasonably sized confidence intervals in our comparisons based on the size of our test sets.

Simulation

In this section we give a concrete example of how a good suggestion can differ from a good prediction, and demonstrate how our methodology can be used to learn a suggestion algorithm that increases the likelihood that suggestions are accepted. We simulate acceptance behavior in order to validate the performance estimates given by TIP.

Parameterized Language Model

Although our methods will allows us to learn and evaluate stochastic suggestion policies of any form, for the purposes of illustration we will use log-linear models with word-level features throughout this paper.

We consider locally-normalized log-linear language models (McCallum, Freitag, and Pereira 2000) of the form

$$p_\theta(s|c) = \prod_{i=1}^{|s|} \frac{\exp \theta \cdot f(w_i|c, w_{[:i-1]})}{\sum_{w'} \exp \theta \cdot f(w'|c, w_{[:i-1]})}, \qquad (3)$$

where $f(w_i|c, w_{[:i-1]})$ is a feature vector for a candidate word given its context. ($w_{[:i-1]}$ is a shorthand for $\{w_1, w_2, \ldots w_{i-1}\}$).

Note that the feature vector can include a wide variety of strong features, such as the conditional log likelihood of w_i under various neural language models (e.g., (Kim et al. 2016)). For the purpose of illustration, we choose a simple set of features. Our first feature is the log likelihood under a base language model $p_0(w_i|c, w_{[:i-1]})$; we follow the approach of (Arnold, Gajos, and Kalai 2016) and use a 5-gram model trained on the Yelp Academic Dataset with Kneser-Ney smoothing, implemented using KenLM (Heafield et al. 2013). We augment that feature with several context-independent characteristics of w_i: a binary feature of if it is long (≥ 6 letters), and a one-hot encoding of its most common POS tag. Note that if the weight vector is zero except for a weight of $1/\gamma$ on p_0, the model reduces to the base language model with "temperature" γ which is often used to control generation from stochastic language models: $\gamma = 1$

corresponds to sampling from the model, while $\gamma \to 0$ corresponds to greedy maximum likelihood generation.

Reference Policy q

When collecting a dataset that we intend to use to evaluate a variety of suggestion models, it is important that a broad range of types of suggestions be offered. Since good suggestions may be far from the most likely predictions, we cannot simply follow the approach of (Arnold, Gajos, and Kalai 2016) and use beam-search to generate maximally likely phrases. However, if the phrases offered are overly exploratory, we risk having few phrases accepted. To balance the extremes of exploration (novelty) and exploitation (obviousness), we choose the reference suggestion policy q to generate suggestions with a temperature of 0.5. We used a fixed reference policy to create a consistent dataset, but future work could refine the reference policy online.

Desirability Model

As Arnold et al. (2016) have shown, people write differently when shown suggestions than a model trained on a text corpus would predict. In this section, we present an illustrative model of suggestion acceptance behavior.

Consider a system that displays 3 suggestions at a time. Suppose further that each suggestion consists of a sequence of words that the author can accept one word at a time; we will discuss the first word acceptances here, modeling the remaining word acceptances proceeds likewise. Let $\{p_j^i\}_{i=1}^3$ denote the likelihood under the predictive model of the first word of suggestion j among the suggestions offered at timestep i; $p_0 = 1 - \sum_{i=1}^3 p_i$ gives the probability of rejecting the suggestions offered. A user behaving according to a pure predictive model would accept the first word of suggestion j with probability p_j^i.[8]

We model the *desirability* of a suggestion by a variable D_j^i that may depend on all the visible words of a suggestion. For example, to model a preference for long words, we can let D_j^i be the number of long words in suggestion s_j^i. We add the desirabilities to the scores of each action:

$$\tilde{p}_j^{(i)} = p_j^{(i)} \exp(D_j^{(i)})/Z^{(i)},\ \tilde{p}_0^{(i)} = p_0^{(i)}/Z^{(i)}$$

$$Z^{(i)} = 1 - \sum_j p_j^{(i)}(1 - \exp(D_j^{(i)})).$$

The net effect is to move probability mass from the "reject" action $\tilde{p}_0$ to suggestions that are "close enough" to what the author actually wanted to say but desirable.

Simulation experiment

We randomly sample 500 suggestion locations from the validation set, where a suggestion location is a sentence up to a specific word, which we cut off and we pretend to retype. We generate three phrases from the reference policy as described earlier, logging their generation likelihoods q. We then allow the simulated author to pick one of the three phrases, subject to their preference.

[8] Assume that the predictive model is fully accurate; our focus is entirely on the difference between prediction and suggestion.

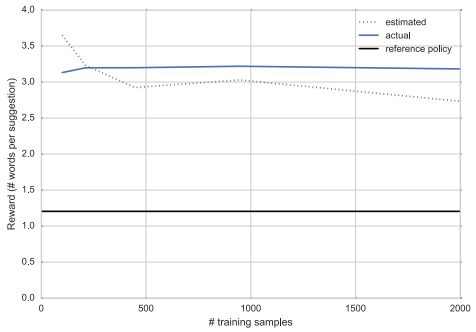

Figure 2: The model is successfully able to learn to make suggestions that are more likely to be accepted by the simulated author who prefers long words. Moreover, the TIP estimates of expected reward tend to be close to the actual expected reward obtained by generating suggestions using the fitted model.

We then use the TIP methodology to find model parameters that are more likely to be accepted. Although the simulator generates three suggestions at once, we treat them as if offered one at a time, ignoring the interaction. Given a training data set $\mathcal{D} = (c^t, s^t, q^t, r^t)_{t=1}^T$, we optimize the reward-augmented likelihood (Gimpel and Smith 2010) with TIP w.r.t. θ:

$$\sum_{t=1}^T \hat{r}_\theta^t = \sum_{t=1}^T r^t \min(p_\theta(s^t|c^t)/q^t, M). \quad (4)$$

Eq. (4) is a concave function and can be maximized by an off-the-shelf solver such as gradient ascent. Note that eq. (4) is related to the risk model introduced in (Gimpel and Smith 2010), while the learning protocol is different.

We then evaluate each learned policy on an additional 500 sentences from the test set. We use the fitted suggestion policy to generate suggestions, compute the probability of accepting each suggestion (including number of words) according to the desirability model. We report the expected number of words that the new policy would accept.

Figure 2 shows that we can adapt the model parameters to make suggestions that are more likely to be accepted. Moreover, the TIP estimates are not only useful for learning; they correlate well with the actual improvement.

Dataset of Interactive Suggestion Acceptances

As we have argued, conventional language corpora do not suffice for learning what makes a good suggestion; we must observe how authors respond to actual suggestions. As Arnold et al. (2016) observed, writers use phrase suggestions more than the same text offered as word suggestions. So we offered phrase suggestions to authors in a composition task and recorded which suggestions they accept. We are then able to use the acceptance data to learn models that result in suggestions that are accepted more often.

We use the phrase suggestion interface from Arnold et al. (2016), which augments the standard word-prediction interface with phrase suggestions, which can be accepted one

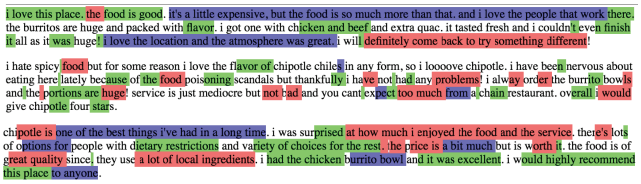

Figure 3: Example reviews from our dataset. A colored background indicates that the word was inserted by accepting a suggestion. Colors correspond to slots on the suggestion bar (left, middle, or right), so consecutive words with the same color were inserted as part of a phrase. Some participants accepted suggestions frequently, others rarely.

# accepted	0	1	2	3	4	5	6
count	27,859	1,397	306	130	91	68	107

Table 1: Our dataset includes 29,958 suggestions made by the system during typing. Authors accepted at least one word of 2099 suggestions (7%), and at least 2 words in 702 suggestions (2.3%). In total, 3745 out of 5125 words in the corpus were entered using suggestions. These acceptance rates are comparable with those observed in other work (e.g., 8% word acceptance in the best condition in (Quinn and Zhai 2016).

word at a time. We refer the reader to that prior paper for further details on the interface. As described above, instead of their beam-search phrase generation procedure, we use our reference policy with a temperature of 0.5. We generate 6-word suggestions after every keystroke.

We recruited 74 workers through MTurk to write reviews of *Chipotle Mexican Grill* using the phrase suggestion interface. Based on pilot experiments, Chipotle was chosen as a restaurant that many crowd workers had dined at.

User feedback back was largely positive, and users generally understood the suggestions' intent. The users engagement with the suggestions varied greatly – some loved the suggestions and their entire review consisted of nearly only words entered with suggestions while others used very few suggestions. Several users reported that the suggestions helped them select words to write down an idea or also gave them ideas of what to write. We did not systematically enforce quality, but informally we find that most reviews written were grammatical and sensible, which indicates that participants evaluated suggestions before taking them.

The dataset contains 74 restaurant reviews typed with phrase suggestions. The mean word count is 69.3, std=25.70. We are providing the data in the form of timestamped logs of what suggestions were offered and what action the author took. We also provide code for aggregating the logs into a table of what suggestions were made and how many words of each suggestion were accepted. Table 1 reports how many words were accepted.

Learning to Suggest Phrases Using the Dataset

We then demonstrate using the TIP methodology to learn to suggest phrases using this dataset. We maximize the estimated expected reward (with M fixed at 10 for simplicity), and evaluate the performance of the learned parameters on held-out data using 5-fold cross-validation. Figure 4 shows that while the estimated performance of the new model does vary with the M used when estimating the expected reward, the relationships are consistent: the reference policy (temperature 0.5) outperforms a purely predictive stochastic policy (temperature 1.0), and the fitted model consistently gets higher expected reward. The fitted model weights suggest that the workers seemed to prefer long words and pronouns, and eschewed punctuation.

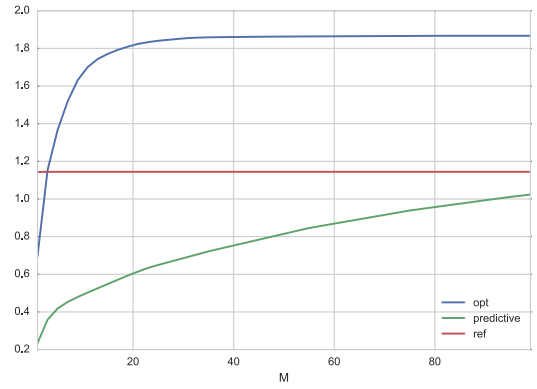

Figure 4: The fitted model consistently improves expected reward (averaged across 5 validation folds), regardless of the M used in estimation. The two baselines shown are the reference policy under which the suggestions were originally offered, and the predictive language model.

Conclusion

We propose the challenge of learning to suggest phrases to authors interactively during the composition process. We provide a dataset of compositions written with interactive suggestions generated from a reference policy. This dataset can be used to design and evaluate suggestion algorithms. We present a simple log-linear suggestion algorithm and demonstrate in simulation and practice that it can be trained to generate suggestions that are more likely to be accepted.

One limitation of our work is that our task of crowdsourced reviews of a single restaurant may not be representative of other tasks or populations of users. However, the predictive language model is a replaceable component of our approach, and a stronger model that incorporates more context (e.g., (Sordoni et al. 2015)) could improve our baselines and extend our approach to other domains. Also, our methodology cannot evaluate a maximum likelihood generation policy because it is not stochastic.

Several natural questions are left for future work. First, how can one design better language models that increase the reward even beyond what we have done with the simple language model features used here? Second, in a system which is capable of offering more than one suggestion, how should one select a menu of suggestions? (Recall that for simplicity

we ignored the interaction between multiple suggestions.)

Finally, our current methodology does not distinguish between suggestions that were accepted for convenience and those that were accepted for their insight or creativity. Some suggestions may in fact bias the user, e.g., a user who intended to write a negative review may be tempted to accept a suggestion that favors the restaurant because it is grammatical and colorful. Other suggestions may be inspiring even if they are not accepted. An interesting open question is how to make suggestions that improve the quality of writing with inadvertently biasing the user. Such work could fit into the TIP framework in this paper by using measures of quality, not just acceptance, as the reward.

References

Arnold, K. C.; Gajos, K. Z.; and Kalai, A. T. 2016. On suggesting phrases vs. predicting words for mobile text composition. In *Proceedings of UIST '16*.

Bickel, S.; Haider, P.; and Scheffer, T. 2005. Learning to complete sentences. In *Machine Learning: ECML 2005*. Springer. 497–504.

Chen, M.-H.; Huang, S.-T.; Hsieh, H.-T.; Kao, T.-H.; and Chang, J. S. 2012. Flow: a first-language-oriented writing assistant system. In *Proceedings of the ACL System Demonstrations*, 157–162.

Gimpel, K., and Smith, N. A. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Guzman-Rivera, A.; Kohli, P.; Batra, D.; and Rutenbar, R. A. 2014. Efficiently enforcing diversity in multi-output structured prediction. In *AISTATS*.

Heafield, K.; Pouzyrevsky, I.; Clark, J. H.; and Koehn, P. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 690–696.

Ionides, E. L. 2008. Truncated importance sampling. *Journal of Computational and Graphical Statistics* 17(2):295–311.

Jiang, J.-Y.; Ke, Y.-Y.; Chien, P.-Y.; and Cheng, P.-J. 2014. Learning user reformulation behavior for query autocompletion. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, 445–454. New York, NY, USA: ACM.

Johnson, M.; Schuster, M.; Le, Q. V.; Krikun, M.; Wu, Y.; Chen, Z.; Thorat, N.; Viégas, F. B.; Wattenberg, M.; Corrado, G.; Hughes, M.; and Dean, J. 2016. Google's multilingual neural machine translation system: Enabling zero-shot translation. *CoRR* abs/1611.04558.

Kannan, A.; Kurach, K.; Ravi, S.; Kaufmann, T.; Tomkins, A.; Miklos, B.; Corrado, G.; Lukacs, L.; Ganea, M.; Young, P.; and Ramavajjala, V. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. M. 2016. Character-aware neural language models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Kneser, R., and Ney, H. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Li, L.; Chu, W.; Langford, J.; and Wang, X. 2011. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the International Conference on Web Search and Data Mining (WSDM)*, 297–306. ACM.

Liu, T.; Zhou, M.; Gao, J.; Xun, E.; and Huang, C. 2000. Pens: A machine-aided english writing system for chinese users. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*.

Mani, I. 2001. *Automatic Summarization*, volume 3 of *Natural Language Processing*. Amsterdam/Philadelphia: John Benjamins Publishing Company.

McCallum, A.; Freitag, D.; and Pereira, F. 2000. Maximum entropy Markov models for information extraction and segmentation. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Mikolov, T., and Zweig, G. 2012. Context dependent recurrent neural network language model. In *SLT*, 234–239.

Nandi, A., and Jagadish, H. 2007. Effective phrase prediction. In *Proceedings of the 33rd international conference on Very large data bases*, 219–230. VLDB Endowment.

Quinn, P., and Zhai, S. 2016. A Cost-Benefit Study of Text Entry Suggestion Interaction. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* 83–88.

Rambow, O.; Bangalore, S.; and Walker, M. 2001. Natural language generation in dialog systems. In *Proceedings of the first international conference on Human language technology research*, 1–4.

Reiter, E.; Dale, R.; and Feng, Z. 2000. *Building natural language generation systems*, volume 33. MIT Press.

Roark, B.; Saraclar, M.; Collins, M.; and Johnson, M. 2004. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, 47.

Shokouhi, M. 2013. Learning to personalize query autocompletion. In *Proceedings of the Conference on Research and Developments in Information Retrieval (SIGIR)*. ACM.

Sordoni, A.; Galley, M.; Auli, M.; Brockett, C.; Ji, Y.; Mitchell, M.; Nie, J.-Y.; Gao, J.; and Dolan, B. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Swaminathan, A., and Joachims, T. 2015. Counterfactual risk minimization. In *Proceedings of the 24th International Conference on World Wide Web Companion*, 939–941. International World Wide Web Conferences Steering Committee.