CS230

Deep Cuisine Transfer (Natural Language Processing)

Arun Karthikeyan
Department of Computer Science
Stanford University
akarthi2@stanford.edu

1 Abstract

Neural Text Style Transfer (TST) [5] is gaining traction since its inception in 2017. TST aims to control the style attributes of the text while preserving content much similar to Image Style Transfer. In this work, we attempt to apply TST to the realm of Computational Gastronomy [1] where we build a model that can perform style/attribute transfer from one cuisine type to another given the recipe instructions, source cuisine class and the target cuisine class. Figure 1 represents a simplified outline of the work.

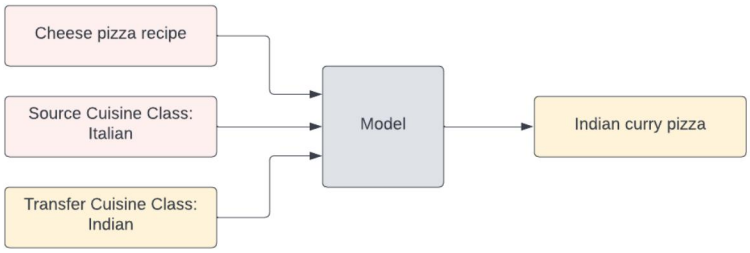

Figure 1: Abstract Architecture

2 Introduction

The application of TST to cooking recipes is a very interesting but novel problem at the time of this writing. Each cuisine has a unique style to its dishes which is often representative of the originating culture. The style could be based on the choice of ingredients, specific cooking methods, utensils used, texture of the dish, the way the ingredients are combined etc. The problem is sophisticated yet interesting because more often than not, foreign dishes are modified and adopted to cater to the taste buds of natives. What's palatable in one cuisine might not be so without modifications for another. That's how a lot of fusion cuisine is born, Indo-Chinese cuisine for example. Cuisine fusion/adoption is a rapidly evolving trend in several countries and Deep Learning can definitely help spearhead the effort here.

CS230: Deep Learning, Spring 2022, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Related work

The author of [1] talks about how digitization has led to a large influx of data into the field of food sciences giving rise to an emerging field - *Computational Gastronomy* which aims to provide insights into the field of Gastronomy based on the application of data-driven strategies. [4] provides an understanding of several different aspects of computational gastronomy including *Food Pairing* and *Novel Recipe Generation* among others, it also provides a brief survey of literature pertaining to these topics. A very helpful work for this project is [5] which is a survey of TST in the recent years. And another interesting piece of work related to this problem is [6] where the authors employ TST to transfer reviews between positive and negative styles while preserving the content. In [7], the authors have presented a novel recipe generation and evaluation system based on generative language models which might be helpful for this project. The authors of [2] have discussed about how they used a Named Entity Recognizer (NER) to extract food entities from their RecipeNLG dataset and feed the extracted entities to a fine tuned GPT-2 language model [9] to generate novel recipes. Another interesting work is [11] where the authors try to decode the ingredients of a recipe based on its image.

4 Dataset and Features

The dataset requirement for this problem is to have recipe-instructions and labeled cuisine region/style that it maps to. Most available recipe datasets [12] either have an *Ingredients:CuisineStyle* mapping or *Ingredients:Instructions* mapping but what we really want is the *Instructions:CuisineStyle* mapping. On first look it might look like we could find some way to join these two and use them, or so we thought originally but its a much harder problem given that they are from different sources. After conducting an extensive search, we found 2 promising sources of curated data CulinaryDB [3] and RecipeDB [10]. RecipeDB specifically had exactly what we needed (and more), it had all the 3 *Instructions:Ingredients:CuisineStyle* fields per row with a size of about 100k recipes. It's available under creative commons license [8]. However, it was hosted online but a direct download link isn't available. Tried contacting the authors but couldn't get a response so the only option left was to crawl the hosted data and construct the dataset ourselves.

4.1 Web Crawler Challenges

The WebCrawler implementation wasn't straight-forward, we had to cross the following hurdles.

1. The recipe region and instructions were available in nested pages and needed separate requests for crawling.
2. Crawling the nested pages with recipe instructions required deeper inspection into the request "post" payload to construct the correct request.
3. The no. of requests to be made was in the order of tens of millions because of nesting, and required batch async requests implementation to speed up the crawling process.
4. Data join business logic was required after scraping the base-pages and nested-pages to get required view.

After writing and testing the web scraper, the scraping part alone took over 20 hours due to server qps throttling for hosted RecipeDB data. But it was all worth it given this data is the backbone of this project. Here's some exploratory analysis on the crawled dataset.

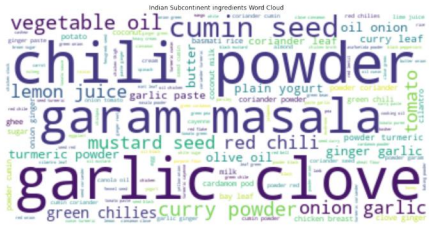

Figure 2: Indian Ingredients Word Cloud

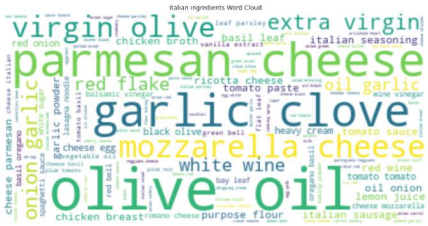

Figure 3: Italian Ingredients Word Cloud

5 Methodology

For simplicity, we reduce our problem from being able to transfer between multiple cuisine classes to just between 2 cuisines that are disparate in nature. Based on ingredient similarity analysis, Italian and Indian is found to be a good choice, Figures 2 and 3. The survey from [5] helped assimilate the TST domain and helped lay the foundation for the model architecture. TST is broadly classified into two application areas, TST on Parallel data where there is labeled-data available across multiple styles with the same content and non-parallel data where only mono-corpora labeled data is available. This project falls in the realm of Non-Parallel TST. Specifically, we use the *Disentanglement* approach where we (i) Encode the text into its latent representation containing the style *s* and content *c* vectors (ii) Control the latent representation to remove all style attribution without tampering the content representation. (iii) Decode the latent representation by supplying a new target style s'. Figure 4 should give an overview of this approach.

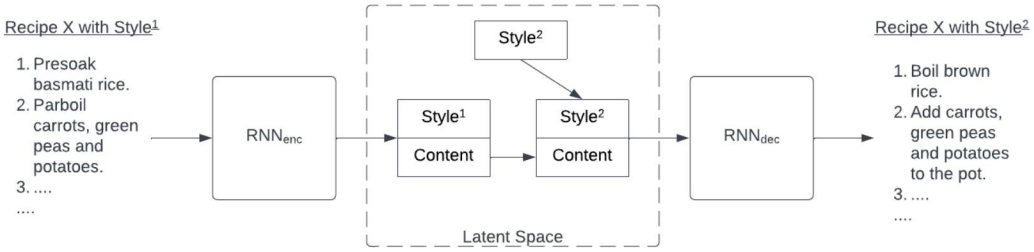

Figure 4: Latent Disentanglement Architecture

We have followed [6] where the authors have applied disentangled representation learning to transfer positive reviews to negative and vice-versa on the yelp dataset, at a high level they rely on autoencoders to learn the latent-representation. The network is incentivized to learn this latent representation based on separate style and content oriented losses similar to Image Style Transfer.

5.1 Auto Encoder for Latent Representation

We use a sequence to sequence GRU based deterministic autoencoder to map the input recipe instructions to a latent space $L \Rightarrow [S; C]$ where S is the encoding of the cuisine style and C is the encoding of the recipe instructions content. The goal is to ensure that the (i) style vector doesn't encode content information, (ii) the content vector doesn't encode style information while (iii) both style and content information completely capture the input recipe instructions. To achieve this we work with a combination of multi-task loss and adversarial loss in addition to the encoder-decoder loss.

$$J_{overall} = J_{auto-encoder}(\theta_{AE}) + J_{style}(\theta_{AE}, \theta_S) + J_{content}(\theta_{AE}, \theta_C)$$

Note that the autoencoder here also has a decoder component which essentially learns to re-generate the input recipe instruction given the content and style encoding.

5.1.1 Loss function for style encoding

There are two parts here, the first one is to ensure that style encoding truly captures the style information, this is achieved by adding a multi-task component to the current model that acts as a style classifier. In this case we will use the binary cross entropy loss (remember that we have only 2 output classes for style - Indian & Italian), the parameters of this style classifier are denoted by θ_S. The gradients from this loss will backpropagate to update θ_{AE} as well. The second part is to ensure that the content encoding doesn't capture any style information. We achieve this by training an adversarial classifier that deliberately discriminates the true style label given the content encoding, The intermediate adversarial classifier's parameter is fixed while we update the content part of θ_{AE} to ensure the content encoding can't discriminate style, for this to be true we would ideally want the

output of this adversarial classifier to be uniformly distributed across both all styles regardless of the input content encoding.

$$J_{style}(\theta_{AE}, \theta_S) = \lambda_{mul(s)} J_{mul(s)}(\theta_{AE}, \theta_S) - \lambda_{adv(s)} J_{adv(s)}(\theta_{AE})$$

5.1.2 Loss function for content encoding

Similar to the style encoding we add a new multi-task component that ensures the content encoding truly captures the content of the input recipe instructions, we use Bag-of-Words (BoW) features to represent the content here. The categorical cross-entropy loss is used for this softmax classifier. There is also an adversarial classifier here that ensures that the style encoding is not able to predict the BoW features of the content. The gradients from both these classifiers are used to update θ_{AE}.

$$J_{content}(\theta_{AE}, \theta_C) = \lambda_{mul(c)} J_{mul(c)}(\theta_{AE}, \theta_C) - \lambda_{adv(c)} J_{adv(c)}(\theta_{AE})$$

Note that we processed the input recipe instructions to get rid of stop-words and cuisine-style specific words before building BoW features. The cuisine-style specific words were crawled from the web (various Indian and Italian cuisine lexicons).

5.2 Style Transfer

The auto-encoder from the previous section allows us to encode the input recipe instructions into its latent space which encompasses the style and content encoding while the decoder component generates the input sentence. The idea is to perform style transfer by injecting the desired style into the style encoding while keeping the content encoding fixed. We can obtain an empirical estimate of any style-category by averaging the style encoding of all examples in that category.

$$Enc(S_T) = \frac{\sum_{i \in S_T} s_i}{\#S_T}$$

Feeding the desired $Enc(S_T)$ along with output content encoding into the decoder should yield us the desired style transferred recipe instructions.

6 Implementation and Evaluation Results

The implementation was inspired by the original work of authors from [6] who built a model for text sentiment style transfer on reviews amazon and yelp reviews dataset. We used Adam optimizer for the autoencoder and RMSProp for the discriminators. The initial learning rate was set to 10^{-3}. The word-embedding layer was initialized by Word2Vec gensim package, we used a 300D work embedding. We used a recurrent unit size of 256 for the seq-to-seq model, the style encoding vector size was 8 while the content encoding vector size was 128.

Recipe Name	Indian Style	Generated Italian Style
Prawn Pepper Masala	Heat oil in a saucepan or wok, add chopped green chile and onions, stir fry until golden brown.<EOI> Add curry leaves, ginger garlic paste and dry spice powders. <EOI> slip in the prawns with 50ml water and cook for 3 to 4 minutes over high heat until done. <EOI>	Heat olive pan <EOI> mix mozzarella onion garlic prawn <EOI> Serve hot. <EOI>

The model was able to split the style encoding reasonably well between Indian and Italian recipes, figure 5. Automatic evaluation of text-generation models is not straight forward, in our case we

built an independent style classifier with word-2-vec embedding features to assert the strength of the transferred style in the generated recipe instructions text. The classifier accuracy on the independent validation set for the Indian and Italian labels was 84.5%. However the assertion for generated style-transferred text was only 73%, we believe it will get better if we augment the current model with context-aware attention mechanism since most of the recipe instructions were long sentences. It should also be noted that the independent cuisine classifier can also be tuned to perform better for more accurate evaluations.

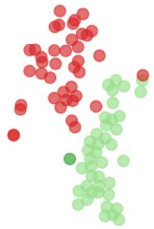

Figure 5: t-SNE plot for sampled Indian and Italian recipe style-encodings

7 Future work

The current model doesn't have attention mechanism built into it to capture long recipe instructions effectively. Since the recipe instructions are usually a combination of several sub-instructions sentences, a context-aware attention based sequence-to-sequence model should be able to perform better than the current plain sequence-to-sequence model. Another direction we could exploring is to use [3] to better group similar ingredients together to reduce the high no. of unique ingredients we're observing (currently about 20k across all recipes), this will help condense the input vocabulary and this particular data source should work since [3] and [10] belong to the same source data.

8 Thanks

I sincerely thank the TAs and the Professors for providing me this exciting opportunity of learning and exploring Natural Language Processing, specifically Text Style Transfer. The knowledge I've gained through this course project was immense, It was a lot of work and I'll most likely explore this further even after the course ends.

References

[1] Sebastian E. Ahnert. "Network analysis and data mining in food science: the emergence of computational gastronomy". In: *Flavour* 2.1 (Jan. 2013), p. 4. ISSN: 2044-7248. DOI: 10.1186/2044-7248-2-4. URL: https://doi.org/10.1186/2044-7248-2-4.

[2] Michał Bień et al. "RecipeNLG: A Cooking Recipes Dataset for Semi-Structured Text Generation". In: *Proceedings of the 13th International Conference on Natural Language Generation*. Dublin, Ireland: Association for Computational Linguistics, Dec. 2020, pp. 22–28. URL: https://aclanthology.org/2020.inlg-1.4.

[3] CulinaryDB. *CulinaryDB: Data Analytics for World Cuisines*. URL: https://cosylab.iiitd.edu.in/culinarydb/.

[4] Mansi Goel and Ganesh Bagler. "Computational gastronomy: A data science approach to food". In: *Journal of Biosciences* 47.1 (Jan. 2022), p. 12. ISSN: 0973-7138. DOI: 10.1007/s12038-021-00248-1. URL: https://doi.org/10.1007/s12038-021-00248-1.

[5] Di Jin et al. "Deep Learning for Text Style Transfer: A Survey". In: *CoRR* abs/2011.00416 (2020). arXiv: 2011.00416. URL: https://arxiv.org/abs/2011.00416.

[6] Vineet John et al. “Disentangled Representation Learning for Non-Parallel Text Style Transfer”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 424–434. DOI: 10.18653/v1/P19-1041. URL: https://aclanthology.org/P19-1041.

[7] Helena H. Lee et al. “RecipeGPT: Generative Pre-training Based Cooking Recipe Generation and Evaluation System”. In: *Companion Proceedings of the Web Conference 2020*. ACM, Apr. 2020. DOI: 10.1145/3366424.3383536. URL: https://doi.org/10.1145%2F3366424.3383536.

[8] License. *Creative Commons*. URL: https://creativecommons.org/licenses/by-nc-sa/3.0/.

[9] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: 2019.

[10] RecipeDB. *RecipeDB: A resource for exploring recipes*. URL: https://cosylab.iiitd.edu.in/recipedb/.

[11] Amaia Salvador et al. *Inverse Cooking: Recipe Generation from Food Images*. 2018. DOI: 10.48550/ARXIV.1812.06164. URL: https://arxiv.org/abs/1812.06164.

[12] Google Dataset search. *Google Recipe Datasets*. URL: https://datasetsearch.research.google.com/search?src=0&query=cooking+recipe.