# Linguistic Preprocessing for Attribute Transfer

Karishma Mandyam

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Bachelors in Computer Science and Engineering

University of Washington

2019

Reading Committee:

Noah A. Smith, Chair

Program Authorized to Offer Degree:
UW CSE B.S. Program

## ACKNOWLEDGMENTS

Thank you to Noah Smith, who inspires me to always keep learning, ask the right questions, and think outside the box.

A special thank you to Swabha Swayamdipta without whom this project would not have been possible. Thank you for always encouraging me to be inquisitive and work smarter. In particular, thank you for teaching me how to approach research with a data-driven perspective.

Thank you to those who taught me the fundamentals of Natural Language Processing, especially Yejin Choi whose classes were so instrumental in building my confidence. Also, Yangfeng Ji, who answered all my questions on neural networks when I started working on research.

Finally, thank you to members of the ARK Undergraduate group: Tam Dang, Deric Pang, Nelson Liu, Aishwarya Nair, Michael Zhang, Ethan Chau, Amy Shah, and Leo Liu. Our weekly discussions have been one of the greatest sources of learning during these past two years.

## TABLE OF CONTENTS


Page

Chapter 1: Introduction . . . . . 1

Chapter 2: Problem Setup . . . . . 3
- 2.1 Motivation . . . . . 3
- 2.2 DELETE . . . . . 4
- 2.3 RETRIEVE . . . . . 4
  - 2.3.1 Sentence Similarity . . . . . 4
- 2.4 GENERATE . . . . . 5

Chapter 3: Syntax-Based Attribute Deletion . . . . . 6
- 3.1 $n$-gram Deletion . . . . . 6
- 3.2 Syntax-Based Deletion . . . . . 7
- 3.3 Unigram Deletion . . . . . 8

Chapter 4: Training and Experimental Setup . . . . . 9
- 4.1 Auto-Encoder Training . . . . . 9
- 4.2 Implementation Details . . . . . 10
- 4.3 Training Parameters . . . . . 10
- 4.4 Dataset . . . . . 11

Chapter 5: Experimental Results . . . . . 12
- 5.1 Evaluation . . . . . 12
- 5.2 Threshold Experiments . . . . . 12
- 5.3 Salience Threshold and Dev Perplexity . . . . . 13

Chapter 6: Analysis . . . . . 15
- 6.1 Deletion . . . . . 15
- 6.2 Delete-Only vs. Delete-Retrieve . . . . . 16

Chapter 7: Related Work . . . . . 19

Chapter 8: Conclusion and Future Directions . . . . . 20
8.1 Future Work . . . . . 20

University of Washington

**Abstract**

Linguistic Preprocessing for Attribute Transfer

Karishma Mandyam

Chair of the Supervisory Committee:
Professor Noah A. Smith
Paul G. Allen School of Computer Science and Engineering

In this thesis we explore controlled text generation, specifically regarding attribute transfer. Our contributions build off the work of Li et al. (2018), who describe a pipeline to transform a sentence exhibiting a source attribute into a sentence exhibiting a target attribute by deleting source attribute markers and carefully selecting target attribute markers. We contribute a method of deleting attribute markers using constituency parses. We explore the effect of this new deletion method on two generation models, Delete-Only and Delete-Retrieve, both of which are described in Li et al. (2018). Our method outperforms the $n$-gram deletion method prescribed by Li et al. (2018) in the Delete-Retrieve model and achieves similar results in the Delete-Only model.

# Chapter 1

# INTRODUCTION

Natural language generation has resurfaced as a prominent application in natural language processing. Text can be automatically produced to summarize long documents, knowledge bases, and even for automating product reviews. The focus of this thesis is on a particular flavor of text generation, called attribute transfer, which involves rewriting text with a single change—converting one attribute of the source text into a desired attribute. Attribute transfer is an example of controlled text generation, since most of the text is required to remain unchanged, except for the attribute. These attributes can either be semantic, such as sentiment or stylistic, such as formality.

This thesis builds on the work of Li et al. (2018) who describe a pipelined method for attribute transfer by editing the source sentence to produce the target sentence. The setting considered is devoid of parallel data. Instead of following an end-to-end procedure for attribute transfer, they break the process down into three steps:

- **DELETE:** Remove the source attribute markers from the original text.
- **RETRIEVE:** Fetch attribute markers from the target corpus.
- **GENERATE:** Construct the target sentence, using the source sentence and the retrieved attribute markers.

All three modules influence the quality of the generation; especially important is the DELETE module affects the RETRIEVE module, both of which determine what gets generated. Evaluation is done on a held-out test set using automated methods such as BLEU Papineni et al. (2002).

This thesis centers around the key observation that the attribute of a sentence manifests most prominently in phrases of text which correspond to syntactic constituents. For example, when identifying the phrases in a sentence which give the sentence a positive attribute, we examine phrases with adjectives or adverbs. Examining linguistic structure gives us a better idea of how different parts of a sentence interact with each other, which in turn, helps us narrow down the scope when searching for attribute markers.

In this thesis, we explore what makes a good attribute marker and how we can more effectively search for these phrases within a given sentence. We propose a method to identify potential attribute markers based on constituency parses and choose the best ones based on experimentally determined scoring thresholds.

# Chapter 2

# PROBLEM SETUP

This thesis builds on the attribute transfer approach of Li et al. (2018), where the focus is on constraint-based text generation from an unlabeled corpus. The setting involves access to two non-parallel text corpora exhibiting a source and target attribute, during training. Generally, the relationship between a source and target attribute would be that the former is either the opposite of the latter or is different from the latter. For example, the training corpus may involve two subcorpora—one containing reviews exhibiting positive sentiment and another with negative sentiment; another example involves subcorpora with humorous text vs. romantic text. Given a sentence $s$ exhibiting the source attribute, the goal is to transform it into a sentence exhibiting the target attribute while retaining the content of the original sentence. For example, given source and target attributes of positive and negative respectively, and the sentence "the food was delicious", we might produce a sentence such as "the food was terrible". To accomplish this, Li et al. (2018) split the generation process into three modules: DELETE, RETRIEVE, and GENERATE.

## *2.1 Motivation*

The motivation for why Li et al. (2018) split the attribute transfer process into these three different steps comes from the fact that there is no availability of a parallel, aligned corpus. In other words, for two given attributes, the training data does not include examples of the same sentence exhibiting different attributes. This prevents us from using only an encoder decoder model, a popular technique used to generate sentences of variable length. The DELETE and RETRIEVE modules described below allow for us to construct input to a standard encoder decoder model, by splitting up a sentence into portions which exhibit the attribute and portions which do not.

Furthermore, the DELETE, RETRIEVE, GENERATE processes describe a form of constrained text generation. This method combines the effectiveness of neural networks with important pre-processing elements, which allow for a balance between the sometimes un-restrained behavior of

neural generation and the structure of non-neural methods. This method also permits us to explore how different syntactic pre-processing methods affect neural generation and especially how well neural models adhere to the structure of pre-processed inputs they receive.

### *2.2* DELETE

Given a sentence $s$ exhibiting the source attribute $a_s$, the goal of the DELETE module is to remove all $n$-grams from $s$ which contribute to the source attribute. The intuition behind this approach lies in the notion that any given sentence exhibiting an attribute can be split into words which exhibit the attribute and words which do not. We refer to the removed words as the source attribute markers. Once the attribute markers are removed, we are left with just the content words of $s$, which should not exhibit any of the source attribute.

### *2.3* RETRIEVE

Recall that the aim of attribute transfer is to transform a sentence exhibiting one attribute (source attribute) into the same sentence exhibiting a different attribute (target attribute). The purpose of the retrieve module is to find attribute markers of the target attribute which fit well into the content words of the input sentence $s$. To accomplish this, we scan through the target attribute's training corpus searching for a sentence similar to $s$. Prior to scanning through the target attribute training corpus, both the source and target attribute training examples have been split into content and attribute marker portions as a result of the delete module. When scanning through the target attribute training corpus, we search for a sentence $t$ with similar content words to the content words of the input sentence $s$. We then retrieve the attribute markers of $t$. The intuition behind this approach is that since the content words of $t$ match the content words of $s$ and $t$'s attribute markers exhibit the target attribute, then those attribute markers must fit well with the content words of $s$.

#### *2.3.1 Sentence Similarity*

For quantifying similarity between content words, Li et al. (2018) recommend any form of word similarity measure such as Euclidean distance or TF-IDF. The latter is an approach which measures how relevant a term is to a document in relation to the term frequency (TF) and inverse document

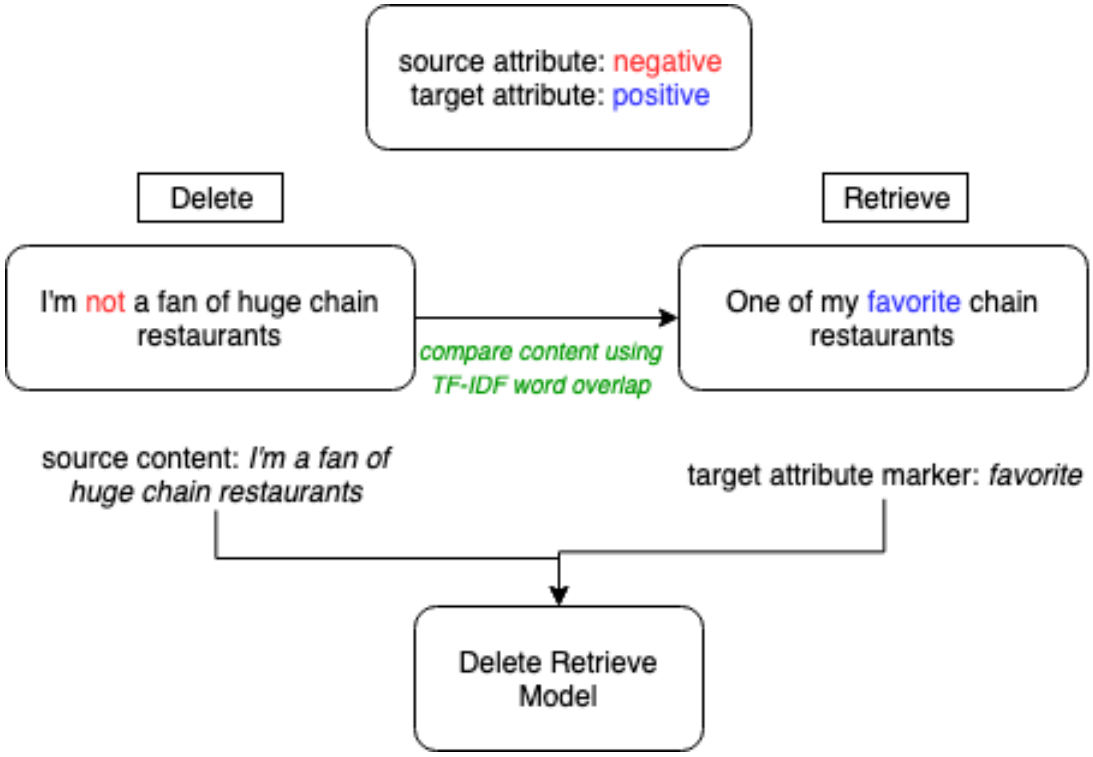


Figure 2.1: The outputs of the retrieve module on a sentence from the YELP dataset

frequency (IDF). The more common a word is in a document or the fewer documents a word appears in, the higher the TF-IDF score. Li et al. (2018) use TF-IDF weighted word overlap to compute similarity between content words.

### *2.4* GENERATE

Li et al. (2018) ultimately recommend two methods of neural text generation, both of which are modifications of an encoder-decoder method. The DELETE-ONLY model skips the RETRIEVE section from above and aims to generate text given the content words from the original sentence $s$ and a learned encoding of the target attribute. The DELETE-RETRIEVE model generates text using the content words from the $s$ and the retrieved target attribute markers. The goal of these approaches is to stitch the outputs of the pre-processing together into a grammatical sentence.

# Chapter 3

# SYNTAX-BASED ATTRIBUTE DELETION

Recall that the goal of the DELETE module is to remove phrases from the input sentence which exhibit the source attribute. Our contributions focus mostly on proposing a syntax-based deletion method which we describe in the following sections. The goal is to take advantage of syntactic structure to determine which phrases in a sentence are related and exhibit an attribute. Given the finding that attributes correspond to phrases or $n$-grams of phrases, this has the potential of reducing the search space of potential attribute markers, since not all $n$-grams in a sentence are syntactically coherent phrases (and hence semantically coherent). In addition, we choose to experiment with another unigram-based deletion method.

## *3.1 $n$-gram Deletion*

Li et al. (2018) use $n$-grams to identify and delete attribute markers. Li et al. (2018) enumerate all $n$-grams (up to $n$ = 4) in a given sentence $s$, assign each $n$-gram a score for how well it exhibits an attribute and delete $n$-grams with a score greater than a certain threshold. These deleted $n$-grams are the attribute markers for $s$.

### *Salience Scoring*

Scoring an $n$-gram $u$ involves quantifying how well an $n$-gram $u$ exhibits an attribute $v$. Li et al. (2018) refer to this score as the salience. They define the salience of an $n$-gram $u$ with respect to an attribute $v$ as the number of times $u$ appears in the attribute's training corpus divided by the number of times $u$ appears in all other training corpora. In our case, the salience of an $n$-gram with respect to the positive attribute corpus is the number of times the $n$-gram appears in the positive attribute corpus divided by the number of times it appears in the negative attribute corpus. In order to account for cases where $n$-grams do not appear in the corpora, Li et al. (2018) use a smoothing parameter $\lambda$ which we fix to 1. The higher the salience, the more relevant the $n$-gram to the attribute corpus and

therefore, more indicative of how well the $n$-gram exhibits the attribute. Given the salience scores of each $n$-gram, Li et al. (2018) delete the $n$-grams with scores above a threshold $\gamma$. This threshold is a hyper-parameter which we experiment with in further sections.

### *3.2 Syntax-Based Deletion*

The $n$-gram method above requires generating a large number of $n$-grams for each sentence, most of which are not ultimately relevant. We aim to reduce the search space here by focusing on $n$-grams which are syntactically related. Constituency parsing and the resulting phrase structure grammars is an excellent way of determining relationships between different parts of a sentence. We use a constituency parser from AllenNLP to parse each sentence; this has been built using the phrase-based parsing approach from Stern et al. (2017). We propose a method where we consider only certain combinations of nodes in the parse tree to be deleted. This solution produces a set of $n$-grams which is a strict subset of the set of all $n$-grams in the sentence to be deleted.

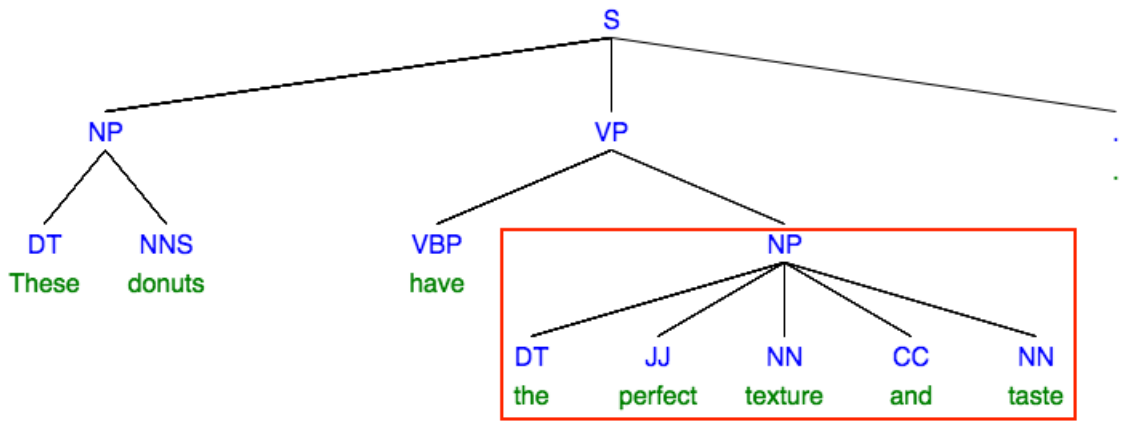


Figure 3.1: A sentence from a review from Yelp and its predicted constituency parse. Terminals are shown in green and non-terminals in blue. With our approach, only the following sub-sequences are considered during deletion: *these donuts, these, donuts, have, the perfect texture and, perfect texture and taste, the perfect texture, perfect texture and, texture and taste, the perfect, perfect texture, texture and, and taste, the, perfect, texture, and, taste*. We do not consider phrases such as *these donuts have the* or *have the perfect* because they do not appear directly as children of the same parent in the parse tree. These phrases would have been enumerated in the $n$-gram approach described earlier. Parse created using Syntax Tree Generator from `http://mshang.ca/syntree/`.

Given a parse tree, we iterate through each node $x$ which contains $m$ children nodes. Each $x$ corresponds to a syntactically relevant sub-sequence of the sentence. For example, in 3.1, the node corresponding to the words "these donuts" represents a noun phrase (NP) which is a sub-sequence of the overall sentence. The leaf of a parse tree always corresponds to individual words such as *these* or *donuts* from 3.1.

We consider uni-gram and bi-gram combinations of these $m$ child nodes with the restriction that each combination may only contain up to four words. In this manner, we can preserve the original n-gram restriction from Li et al. (2018)

We retain a similar scoring scheme to Li et al. (2018) with a salience calculation and respective threshold $\gamma$. The salience calculation here is based on counts of candidates retrieved from the parse based enumeration above.

### *3.3 Unigram Deletion*

We experiment with a third method of deletion based on unigrams. This method serves as a baseline deletion method: it is simple and easy to implement. We delete individual words based on whether their salience score is above a threshold $\gamma$. This is a simplified version of the $n$-gram method employed by Li et al. (2018) and uses the same salience calculation system.

# Chapter 4

# TRAINING AND EXPERIMENTAL SETUP

## *4.1 Auto-Encoder Training*

Training the DELETE-ONLY and DELETE-RETRIEVE models would traditionally require aligned data. However, with this particular setup, we do not have that kind of data. Instead, we follow the procedure of Li et al. (2018) to train the models as auto-encoders. Given a sentence $s$ and a source attribute $src_{attr}$, we train the DELETE-ONLY model to take the content words of $s$ and a learned encoding for the $src_{attr}$ and produce the original sentence $s$. Similarly, for the DELETE-RETRIEVE model, we train the model to take the content words of $s$ and the source attribute markers of the same sentence to produce the original sentence $s$. In the case of the DELETE-RETRIEVE model, we randomly perturb the input attribute markers. This prevents the model from learning to naively append or stitch the content and attribute markers together. We summarize the training vs. testing procedure for the DELETE-ONLY model in 4.1

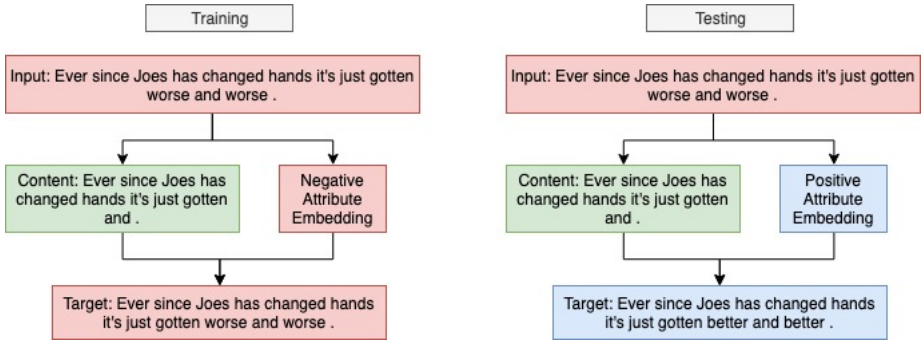


Figure 4.1: Here we demonstrate the difference between training and testing time with the DELETE-ONLY model. During training, our goal is to reconstruct the original sentence from its content and attribute constituents. During testing, we take as input a sentence of a source attribute and attempt to produce a sentence of a different attribute by changing the attribute encoding.

### *4.2 Implementation Details*

The publicly available codebase[1] by Li et al. (2018) is written using the Theano library [2]. For ease of use, we built on a PyTorch-based[3] implementation recommended by the authors[4]. This repository provides implementations for the DELETE-ONLY and DELETE-RETRIEVE models from Li et al. (2018) with the following differences:

- Rather than using single layer GRUs, we use two layer LSTMs
- During train time, we retrieve similar attribute markers using word overlap instead of word-edit distance
- During decoding, we do not use beam-search with neural language models and instead opt for a greedy approach.

These changes are taken from Pryzant's repository and while we retain the general architecture described by Li et al. (2018), we do not experiment with the effect of changing the decoding method on the outputs. We do generally notice that using an LSTM over a GRU improves outputs slightly.

### *4.3 Training Parameters*

For the most part, we keep the original training parameters from Li et al. (2018). They use single layer RNNs with embedding size 128 and a hidden layer dimension 512 for both the encoders and decoder. We use the same, except with two layer RNNs. The $n$-gram method of deletion uses $n = 4$ and the smoothing parameter $\lambda$ for calculating salience is always set to 1. Parameters for the neural networks are initialized by sampling from a uniform distribution between $-0.1$ and $0.1$ and we use the Adam optimizer with a minibatch size of 256. Li et al. (2018) choose to fix salience thresholds for their models, but we are interested in experimenting with several different thresholds, which we explore in our experiments.

---

[1] https://github.com/lijuncen/

[2] http://deeplearning.net/software/theano/

[3] https://pytorch.org/

[4] Reid Pryzant's implementation: https://github.com/rpryzant/delete_retrieve_generate

### *4.4 Dataset*

Li et al. (2018) use three datasets to evaluate their models: Yelp Reviews, Amazon Reviews, and a Captions dataset. We elect to evaluate our models only on Yelp data [5]. The Yelp dataset is comprised of business reviews labeled with either a positive or negative sentiment.

[5] https://www.yelp.com/dataset/

# Chapter 5

# EXPERIMENTAL RESULTS

The main goal of this thesis is to compare how different deletion methods affect the outputs of both our attribute transfer models. We compare the outputs of different deletion methods and also explore how the salience threshold affects the quality of generated text.

## *5.1 Evaluation*

We evaluate the generated text from the Delete-Only and Delete-Retrieve models using BLEU scores. Li et al. (2018) collected human annotated examples of input sentences of positive sentiment and target sentences of negative sentiment. We use the latter as a gold standard when calculating BLEU scores. We use the corpus BLEU score from NLTK [1], which differs from the BLEU calculations in the original paper. Li et al. (2018) use a custom script for calculating BLEU scores. As a result, we cannot directly compare our outputs with those of Li et al. (2018). Rather, the purpose of this thesis is to experiment with with syntax based deletion as opposed to $n$-gram based deletion.

Note that during testing time, we evaluate performance using BLEU score. However, during training time, we evaluate performance on the development dataset using perplexity. The development dataset consists of examples similar to those in training (non-parallel) while the testing dataset consists of parallel data so that we can evaluate BLEU score.

## *5.2 Threshold Experiments*

One of the goals of this thesis is to explore how salience threshold affects the quality of predictions. We ran both the Delete-Only and Delete-Retrieve models with unigram, $n$-gram and parse deletion methods on a variety of different thresholds to find the best salience threshold for each combination of model and deletion method.

---

[1] https://www.nltk.org/_modules/nltk/translate/bleu_score.html

| Delete Only Model | | |
|---|---|---|
| Deletion Method | Salience Threshold | BLEU Score |
| Unigram | 144 | 0.577 |
| Parse | 233 | 0.567 |
| N-Gram | 468 | 0.570 |

| Delete Retrieve Model | | |
|---|---|---|
| Deletion Method | Salience Threshold | BLEU Score |
| Unigram | 117 | 0.540 |
| Parse | 969 | 0.534 |
| N-Gram | 468 | 0.472 |

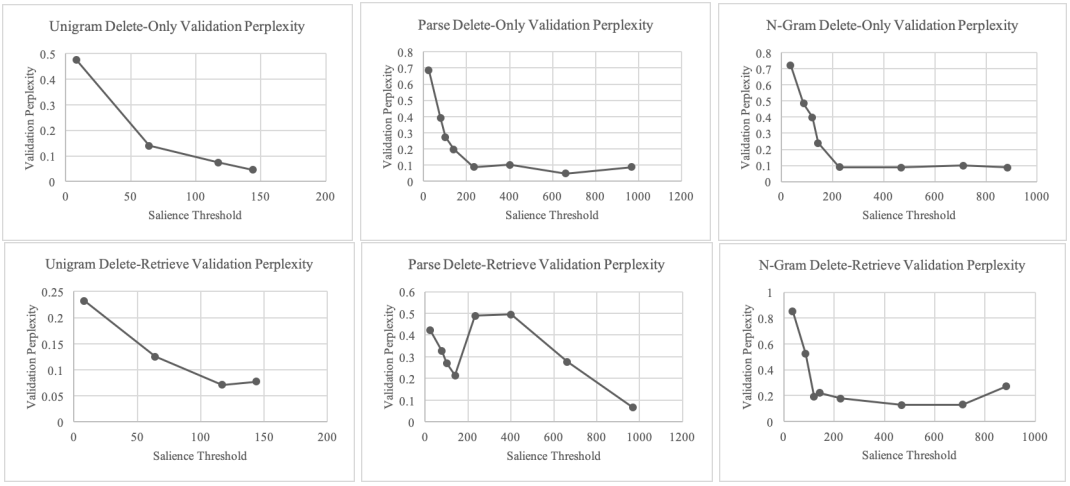


Figure 5.1: Salience threshold experiments show that the models perform best at thresholds significantly higher than those recommended by Li et al. (2018)

Notably, the Delete-Only model performs relatively better than the Delete-Retrieve model on this dataset. Unigram deletion performs the best overall, by a small margin. Parse based deletion performs significantly better than N-Gram deletion on the Delete-Retrieve model. We determined the best thresholds by randomly sampling several thresholds within the range (0, 1000).

### *5.3 Salience Threshold and Dev Perplexity*

In general, we notice that salience threshold has a significant effect on the log perplexity on validation data.

| Deletion Method | 5 | 10 | 15 | 25 | 100 |
|---|---|---|---|---|---|
| unigram | 2959 | 1814 | 1185 | 603 | 106 |
| parse | 69670 | 31872 | 16839 | 8604 | 1378 |
| ngram | 228421 | 96493 | 47203 | 22907 | 2987 |

Figure 5.2: Increasing the salience threshold decreases the size of the attribute vocabulary. The unigram attribute vocabulary is much smaller than both the parse and the $n$-gram vocabularies. Likewise, the parse vocabulary is significantly smaller than the $n$-gram vocabulary. Though we randomly sampled thresholds in our experiments, we report these numbers on specific threshold to show how attribute vocabulary size can change with threshold.

Li et al. (2018) recommend a relatively low salience threshold (15 for the $n$-gram deletion method). However, we find that higher thresholds tend to perform better. A higher salience threshold limits the size of the attribute vocabulary, which encourages the model to retain only higher quality attribute markers. At the same time, a higher salience threshold discourages the model from deleting phrases in the original sentence, which puts more pressure on the retrieve module.

# Chapter 6

# ANALYSIS

Calculating a BLEU score is one way to measure the quality of generated text. However, this measure penalizes sentences which may be grammatical and valid but are far from the human evaluated outputs provided by Li et al. (2018). We can subjectively evaluate the performance of each combination of deletion method and model by taking a closer look at the text generated. The examples referenced in the following sections were hand-picked examples.

## *6.1 Deletion*

Separating attribute transfer into three modules (delete, retrieve, generate) allows us to analyze the intermediate outputs of each module. Of these three modules, we can start by focusing on the deletion module since the attribute markers and content chosen here determine what is retrieved and ultimately, what is generated.

6.1 shows three different examples of how each module performed. In the first example, *I called at 6:30 and got the complete brush off*, $n$-gram and unigram deletion methods tend to delete a lot more of the sentence. We notice that generally, the parse method deletes fewer words than the $n$-gram method, mostly because the attribute vocabulary for the former is much smaller than the latter. In the second example, *We went there on a busy holiday weekend and the service was extremely slow*, we see that all three methods focus on the last part of the sentence. The unigram method captures the smallest set of attribute markers, which, in this case, works very well. The unigram method, unsurprisingly deletes smaller portions of text at a time since it considers each word separately. In this case, the entire phrase *was extremely slow* is a valid phrase to delete from the original sentence. 6.2 shows how deleting different amounts of this phrase affects the generate module with the Delete-Retrieve model. It turns out that all three models produce high quality, grammatical, and relevant sentences. However, the $N$-gram deletion method produces a sentence which is arguably most opposite to the original sentence.

| Deletion Method | Deletion Module Output |
|---|---|
| Unigram | I called at 6:30 and got the complete brush off .<br>We went there on a busy holiday weekend and the service was extremely slow<br>It 's not my fave , but it 's not awful . |
| Parse | I called at 6:30 and got the complete brush off .<br>We went there on a busy holiday weekend and the service was extremely slow<br>It 's not my fave , but it 's not awful . |
| N-Gram | I called at 6:30 and got the complete brush off .<br>We went there on a busy holiday weekend and the service was extremely slow<br>It 's not my fave , but it 's not awful . |

Figure 6.1: Sample outputs from the three different kinds of delete modules. Attribute markers are highlighted in blue and what remain are the content words. The outputs of the delete module are the same for both the Delete-Only and Delete-Retrieve models.

In a vast majority of cases, the unigram deletion method does a better job singling out and deleting attributes. When searching for attributes to delete, the unigram model never considers combinations of words as a phrase. If a word is an attribute marker, it gets chosen and removed. If several words in a phrase are attribute markers, then the unigram method will individually delete all of them. The parse and n-gram methods have attribute vocabularies which consist primarily of multi-word phrases, and finding an exact match for those phrases is more difficult than in the unigram case. These multi-word phrases sometimes include content words such as in the phrase, *was horrible*. The unigram method would have included just *horrible*. Deleting just the phrase *horrible* has an additional benefit. In the case of the Delete-Only model, retaining the word *was* gives the generation module extra structure to predict the next word.

### *6.2 Delete-Only vs. Delete-Retrieve*

We can also compare the performance of the Delete-Only model in relation to the Delete-Retrieve model. Figure 6.3 demonstrates this comparison.

Most noticeably, when the delete module removes a larger number of attribute markers (or even removes parts of the content), finding a suitable set of attribute markers in the retrieve module

| Deletion Method | Deletion Module Output | Generation Module Output |
|---|---|---|
| Unigram | We went there on a busy holiday weekend and the service was extremely slow | We went there on a busy holiday weekend and the service was extremely friendly. |
| Parse | We went there on a busy holiday weekend and the service was extremely slow | We went there on a busy holiday weekend and the service was excellent. |
| N-Gram | We went there on a busy holiday weekend and the service was extremely slow | We went there on a busy holiday weekend and the service was quick. |

Figure 6.2: Sample outputs from the three different kinds of delete modules and the resulting generated text. The generated outputs come from the Delete-Only model. The gold standard sentence here is *We went there on a busy holiday weekend and the service was extremely fast.*

| Deletion Method | Deletion Module Output | Retrieve Module Output | Delete-Only Generation | Delete-Retrieve Generation |
|---|---|---|---|---|
| Parse | My fault for not looking more closely , i guess . | very authentic looking bar . | my for not looking more closely , i promise . | my for not looking more closely , i guess i guess. |
| N-Gram | My fault for not looking more closely , i guess . | will go back for more! | my more closely was the best ! | my more search here ! |

Figure 6.3: Delete-Only vs. Delete-Retrieve intermediate outputs on both parse and n-gram deletion. The phrases in blue are the source attribute markers and the phrases in red are the retrieved attribute markers. The gold standard sentence here is *I looked really close*.

becomes harder. Note that the retrieve module depends on a content to content comparison between the source sentence and examples in the target corpus. The more content words or in general, the more phrases we delete from the original sentence, the less likely it is to find a sentence in the target corpus which matches up nicely. We see this flaw in both cases here. The sentences which are retrieved have little to nothing to do with the original ones and the attribute markers associated with the retrieved sentences might even go so far as confusing the Delete-Retrieve model. Perhaps this is why the model struggles to produce grammatical outputs.

# Chapter 7

# RELATED WORK

There have been several attempts to perform attribute transfer on an unaligned corpus (Shen et al., 2017; Zhao et al., 2018; Fu et al., 2018; Melnyk et al., 2017). These attempts focus on separating the attribute markers from the content of a sentence using adversarial methods . Li et al. (2018) point out that these approaches sometimes result in the encoder fooling the discriminator without actually removing any attribute information from the content embedding. Li et al. (2018) also bring up the point that we have prior knowledge that the attribute portions of a sentence will appear in certain areas, and by using the DELETE and RETRIEVE modules, we take advantage of that.

Syntactic structures have been explored for controlled text generation in other recent work. Iyyer et al. (2018) describe a method of generating paraphrases from parses using syntactically controlled paraphrase networks. These networks use constituency parses to control the structure of the generated text. This work shows that incorporating syntax into the generation process produces sentences which follow a specified syntactic template, learned automatically. This provides an interesting balance of control and generation. In this thesis, we explore incorporating syntax into a pre-processing phase because the products of pre-processing can heavily influence the rest of the training pipeline.

# Chapter 8

# CONCLUSION AND FUTURE DIRECTIONS

This thesis outlines a new method of deletion using constituency parses. Through this method we reduce the search space of possible attribute markers in a given sentence and also show that the attribute markers chosen perform either comparably well or better than those produced with $n$-gram deletion. We find that though the models score well in terms of BLEU metrics, they still struggle to produce grammatical outputs, especially the Delete-Retrieve model, which must also learn to incorporate attribute markers from a retrieved sentence in the target corpus. We also explore how salience threshold affects perplexity on the validation data. To a certain point, higher salience thresholds result in more high-quality attribute markers. However, there is a trade-off between preferring high quality attribute markers and not deleting anything from a sentence. In other words, the higher the salience threshold, the less likely a phrase is to be deleted, even if it is truly an attribute marker.

## *8.1 Future Work*

In this thesis, we described methods which used syntax in a non-neural portion of a model split into two non-neural and one neural module. In the future we intend to explore how syntax can be incorporated specifically during generation time. A more generalized question that would be interesting to answer is: how can we control the proportion of content and attribute markers incorporated into the generated text? Once again, we can explore this by examining syntactic cues which might suggest areas to insert or delete words. Incorporating more syntactic information might also help us generate more grammatical sentences.

## BIBLIOGRAPHY

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *AAAI*. 7

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proc. of NAACL*, pages 1875–1885. 7

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, Retrieve, Generate: a simple approach to sentiment and style transfer. In *Proc. of NAACL*, pages 1865–1874. (document), 1, 2, 2.1, 2.3.1, 2.4, 3.1, 3.1, 3.2, 3.3, 4.1, 4.2, 4.3, 4.4, 5.1, 5.1, 5.3, 6, 7

Igor Melnyk, Cicero Nogueira dos Santos, Kahini Wadhawan, Inkit Padhi, and Abhishek Kumar. 2017. Improved neural text attribute transfer with non-parallel data. ArXiv preprint arXiv:1711.09395. 7

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. 1

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proc. of NeurIPS*, pages 6830–6841. 7

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *ACL*. 3.2

Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander Rush, and Yann LeCun. 2018. Adversarially regularized autoencoders. In *Proc. of ICLR*, pages 5897–5906. 7