Sentence Generation using Fan Theories

Project Report

CS585, UMass Amherst, Fall 2016

Submitted By: Tanvi Sahay

Abstract

In this project, performance of different methods for the purpose of predicting the storyline of a television series using fan theories, was analyzed. These methods involved different combinations of input data and generative models and assessment was done using domain knowledge as well as overall coherence of the generated sentences, by way of manual rating. Tools from the Stanford CoreNLP package were used to perform actions such as tokenization, part-of-speech tagging, named entity recognition, co-reference resolution and relation extraction and their performance over the data set was also analyzed. It was found that relation extraction using Stanford OpenIE is a potential method of ridding the data set of all noisy information while retaining the important details. It was also found that N-gram models perform well with such noisy data as compared to other generative models such as Hidden Markov Models and character-level LSTMs.

1 Introduction

Natural Language Generation (NLG) is the field of natural language processing that deals with text generation on the basis of some input data. In the past, NLG systems have been developed for general tasks such as lyrics/poetry generation[3] or image captioning[11] as well as personalized tasks such as the STOP system that generates personalized pamphlets to help people quit smoking. The task of text generation can be though of as printing out a sequence of words and/or phrases where what the possible sequence can be is learned from a training corpus. While sophisticated machine learning techniques such as recurrent neural networks have been developed to take into account the long-term dependencies in this example text and thus generate more human-like sentences, teaching machines to understand and reproduce the human language has been an uphill task and system has still not achieved perfection. Also, since computers are unable to understand the meaning of the text being provided to them and only learn how to generate sentences based on the sequential occurrence of words and phrases, mapping information or domain knowledge of a particular textual database into the generated sentences has been a challenge that has not yet been successfully tackled. Traditionally, generative models are trained with a rich database in order to compensate for their lack of contextual understanding since a large database will contain more phrases and tokens for the model to learn from, which will generally result in more fluent sentences. However, performance of these models on a noisy and sparse database is yet to be analyzed.

Another part of natural language processing that goes hand in hand with text generation is information extraction. Several *open* information extraction(IE) systems such as TextRunner, WOE and Stanford OpenIE are now available to be used with any database and these allow one to extract relational tuples from input sentences, along with performing other tasks such as tokenization, lemmatization, part-of-speech tagging, named entity recognition, co-reference resolution and parse trees preparation. These open systems have been extensively used as pre-processing steps for various tasks as they provide the basic breakdown of a database in a single package and can often improve the performance of the NLP models. A specific way in which the extracted relational tuples and other extrcated information can be utilized is to remove noise from the database and improve the quality of the text being used to train models and generate new sentences.

In this project, the task of sentence generation has been explored for a database made up of fan theories. In the last 20 years, American entertainment, ranging from movies to television shows to comic books to novels, has seen a colossal increase in its fan following, and with this increase, more and more people have begun to keep a story going even when it is on hiatus, giving rise to fan-proposed theories of what the future of a story might be. Three generative models, N-grams, Hidden Markov Models(HMM) and Long-Short Term Memory(LSTM) recurrent neural networks have been explored and their results compared to analyze how they perform for the proposed task. The Stanford OpenIE package has been used to extract relational tuples while the CoreNLP package has been used to tokenize the input text, perform part-of-speech tagging and find the named entities in the text. This information has been utilized in combination

with the N-gram models to propose three different methods of sentence generation and assess how they perform as compared to other baseline models. Frequency of occurrence of each character name from the chosen show has been computed and sentence for the five most famous characters have been generated, with each sentence generation task being seeded with the name of the character. Comparison has been done on the basis of how understandable a sentence is and how much information it conveys. Human subjects with both presence as well as absence of domain knowledge were asked to rate the sentences, in order to obtain an understanding of how well each model maps domain knowledge in its generative methodology.

2 Literature Review

The following section takes a look at some of the previous attempts at the task of sentence generation using a particular database or for a particular purpose.

2.1 Rap Lyric Generator [3]

In this paper, two generative models, linear-interpolate trigram model and linear-interpolated quadgram model have been trained on a data base of rap lyrics and have been used to generate new rap songs. The authors have made use of a rhyming database, to prepare flatfiles of all the word-rhyming word possibilities, to be incorporated into the system while generating lyrics so that whether the last words of two consecutive lines rhyme or not can be checked. The models have been used in two ways, first when no particular input song is given to a model and second, where a single input song is provided. It is found that the linear-interpolated quadgram model outperforms the linear-interpolated trigram model in that the sentences generated are more grammatically correct. Also, it is found that when an input song is provided, the lyrics generated capture the general theme of the song better than when no input song is given, in which case multiple themes seem to mash up with each other. The sentences have been ranked on several manually defined metrics and for each possible sentence, multiple candidates are generated and the candidate with the highest ranking is selected.

2.2 Information Extraction and Text Generation of news reports for a Swedish-English bilingual spoken dialogue system [1]

This paper describes a dialog system which retrieves information from internet news reports related to user queries in Swedish and English and generates their summaries. Lexicon-governed parsing has been performed on the data and parsing output has been matched to a pre-defined event template. Main events are identified with the help of verbal nouns, predicative adjectives and verbs that are not combined with verbal subjects. The additional but less relevant information is identified with the help of tense markers, conditional markers, modal verbs,and time and space adverbs. Once the templates have been filled, grammar components of Swedish and English generate the final text. This bilingual text generation has been shown to be better than text translation since the Swedish translation does not suffer from lexical and syntactic interference.

2.3 A Probabilistic Approach to Text Generation of Human Motions extracted from Kinect Videos [5]

In this paper, text generation has been performed with multimedia information as input. Human motions have been captured using a kinect camera and time series of the collected data has been extracted. After application of several dimensionality reduction procedures, the observed time-series has been stored in a data base with the intermediate representations which correspond to the semantics of human motions. This information has been learned using machine learning, with visual information as the input and intermediate representation as the output. Once the intermediate information has been learned, sentences describing human motion are collected and bi-gram models for each intermediate representation are prepared. For each input time series, an intermediate representation is decided and from the corresponding bi-gram model, the most likely combination of words is selected by applying dynamic programming to the selected bi-gram.

2.4 Probabilistic Text Structuring: Experiments with Sentence Ordering [7]

This paper presents a method of ordering information that is particularly suitable for text-to-text generation tasks. An unsupervised probabilistic model that learns ordering constraints from a large corpus has

been proposed. The model learns which sequences of features are likely to co-occur and makes predictions concerning preferred orderings. Global coherence is obtained by greedily searching through the space of possible orders. Instead of the best possible ordering, simply an acceptable ordering is constructed. The method of evaluating these orderings has been proposed as an automated technique to find the closeness or distance of the proposed orderings from the gold standards which are a collection of orders produced by humans.

2.5 Lessons from a failure:Generating tailored smoking cessation letters[9]

This article talks about STOP, a natural language generation system that generates short tailored smoking cessation letters, based on responses to a four-page smoking questionnaire. This questionnaire consisted of questions on smoking habits and beliefs, previous attempts to quit, current medical problems and so on. Processing was divided into the stages of document planning, micropanning and realization. The document planner worked by classifying smokers into one of 7 categories and then running a high-level category-specific schema that specified which sections and paragraphs should be included in the letter. The category schema also specified the importance of different sections and paragraphs, which influenced their length. It was found that the STOP system did not perform very well and was as effective as a non-tailored letter in helping people quit smoking. The article also discusses how important it is to state and evaluate negative results and failures and emphasizes on understanding the reasons for the failure of a system, which they state is rare in AI related fields.

2.6 Hidden Markov Models suitable for Text Generation[10]

This paper presents the application of Hidden Markov Models for the purpose of text generation in the polish language. Hidden markov models of orders varying form 1 to 7 were trained using a corpus of reference polish text and the models were used to predict the next character in a sequence. For that purpose, the three most probable words were found and one of them was chosen at random. For higher values of k, it was found that performance was good with almost no misspelled words and this was chalked up to the fact that polish words are mostly 6 to 7 characters long. It was concluded that while HMMs can be used to generate text, they still perform poorly in the case of sentence generation.

2.7 Corpus-Guided Sentence Generation of Natural Images[11]

In this paper, a method that predicts the most likely nouns, verbs, scenes and prepositions that make up the core sentence structure in order to properly describe an image has been proposed. This description consists of a noun, a verb, a scene and a preposition that relates the objects to the scene, all of which have been constrained to a predefined set. Using an image as the input, objects are detected using trained algorithms. Given a set of objects in the image denoted by nouns, verbs are determined conditioned on the set of detected nouns. Given the objects and verb, a scene is predicted and given the scene, the preposition that could be associated with it is predicted as well. The most probable quadruple is then predicted using HMM and this quadruple is used to prepare a descriptor sentence.

2.8 Forest-based statistical sentence generation[6]

This paper presents a new approach to statistical sentence generation in which alternative phrases have been represented as packed sets of trees, or forests, and then have been ranked statistically to choose the best one. A ranking algorithm has also been described, which compares the proposed method with simple enumeration or a lattice-based approach. The paper first describes the lattice representation of text that incorporates corpus knowledge and defines forests in terms of a lattice as providing a single label to each unique arc and to each group of arcs in the lattice, thus eliminating the problem of duplication. Using a bigram model to compare the sentences, it has been concluded that forest representation is 3-4 seconds faster than the lattice representation and that this time does not increase with increase in the sentence length.

2.9 Generating Sentences with Recurrent Neural Networks[2]

This paper presents the use of Long-Short Term Memory recurrent neural networks for the purpose of generating complex sequences with long-range structure. The LSTMs have been trained on the Penn Treebank and the Hutter Prize Wikipedia datasets and performance has been found to be at par with the state-of-the-art language models. The prediction network has then been applied to real-valued data through the use of a mixture density output layer and results have been provided on the

IAM Online Handwriting Database. Sentences have been generated using the Wikipedia database and sample handwriting outputs have been generated using the IAM Handwriting database. In both the cases, LSTMs have been found to perform well, showcasing the network's ability to model long-range dependencies. Certain methods to improve the legibility of the handwriting outputs and learn to produce human-like writing biased towards a particular writer have also been presented.

2.10 Automatic Headline Generation for Newspaper Stories[12]

This paper explores the use of Hidden Markov Models for the purpose of automatically generating headlines for English texts. The headlines and stories have been considered joint outputs of a generative model and for an ordered subset of the first N words of the story taken as the headline H, the H that maximizes its probability of generation given a story has been found as the headline. This probability is found by employing the Bayes' theorem, with the help of bigram models for the probability of H and a Hidden Markov Model to find the probability of story given the headline. Certain constraints have been added to make the generated headlines more human-like and morphological variants of words have also been considered while emitting a word for the headline. It has been found that inclusion of the variants, along with the constrains significantly improves the fluency and accuracy of the generated output, as compared to the baseline HMM outputs.

3 Database

For the purpose of this project, fan theories of the television series Game of Thrones were used as the database. Fan theories of a series are theories as speculated by fans of the show about what the future course of the series might be, based on the events that have already occurred. Several websites, public forums and blog posts were scraped through and a database of 156 documents was prepared. Each text document contained one theory, which was either about a single character or explored the interlinked story lines of more than one. Analyzing these theories shows that they depend heavily on what has already transpired in the series and thus require substantial domain knowledge in order to be properly understood. Also, since fan theories are provided by individuals, they tend to be disparate and a database consisting of fan theories can in general be thought of as sparse with low repetition. This means that every character has more than a single theory to their name. This knowledge is important as it directly affects the generated sentences. With more than one possible theory to be learned for each character, every generative epoch gives a different sentence which makes the outputs both interesting and harder to evaluate, since no base 'correct' theory exists to compare the generated sentence to. Another property of this database that needed to be kept in mind when performing the generative task was that the theories contain high noise. Along with a theory, each text document also contains supportive evidence to substantiate that theory. However, for this project, only the sentence(s) providing the theory itself were important. This presented a challenge since now the database contained a lot of text which was redundant and did not need to be present in the corpus. Presence of jargon was also a definitive property of this database. The corpus consisted of several words and phrases which only held meaning in the realm of Game of Thrones and were otherwise nonsensical. This information provides the intuition that domain knowledge will be necessary evaluate how well a model has learned from the training data since to a person without domain knowledge, all the jargon would simply be gibberish.

3.1 Data Pre-processing

Once the data was collected and the corpus was prepared, tokenization revealed the vocabulary to be 2879 and number of tokens to be 20,594. Tokenization was done to resemble the Stanford CoreNLP tokenization as closely as possible to maintain consistency, since POS tagging and NER were both performed using the said package. In order to do this, certain hand-coded rules had to be added to the simple sentence splitting technique of tokenization. Brackets, both opening and closing and punctuations were considered to be separate tokens and the right and left brackets were replaced with the terms $-rrb-$ and $-lrb-$ respectively in order to unite all brackets under the same token. Words with an apostrophe, such as $Arya's$ and $haven't$ were broken into two tokens, with characters before apostrophe presenting one token and characters after apostrophe, including the symbol, presenting a separate token, given as $Arya$ and $'s$. Once split, all tokens were normalized to lower case. Since tasks were performed using Python and its standard encoding scheme left certain utf8 characters unchanged, these had to be hand-converted to their utf8 version. Once the conversion was complete, tokens such as $\backslash xe2\backslash x80\backslash x99ll$ which represents $'ll$ were kept while others such as $\backslash xe2\backslash x80\backslash x9d$ which means " i.e. right double quotation

mark, were ignored. A sample text and its tokenized form after completion of all pre-processing have been shown in figure 1 and figure 2 respectively.

More Names Checked Off Arya Starks Kill List: In Season 3 Arya (Maisie Williams) began keeping a list of people she planned to kill. Much like Nights Watch recruiter Yoren (Francis Magee), who told her about his own kill list prior to his death, the majority of the men and women listed had wronged Arya or her family in some way. At the close of Season 6 Arya crossed a major name off her list Walder Frey (David Bradley), who hosted the infamous Red Wedding during which her brother, Robb Stark (Richard Madden), and mother, Catelynn Stark (Michelle Fairley), were killed.

Figure 1: Sample text

more, names, checked, off , arya, stark, 's, kill, list, in, season, 3, arya, -lrb-, maisie, williams, -rrb-, began, keeping, a, list, of , people, she, planned, to, kill, ., much, like, night, 's, watch, recruiter, yoren, -lrb-, francis, magee, -rrb-, who, told, her, about, his, own, kill, list, prior, to, his, death, the, majority, of , the, men, and, women, listed, had, wronged, arya, or, her, family, in, some, way, ., at, the, close, of , season, 6, arya, crossed, a, major, name, off , her, list, walder, frey, -lrb-, david, bradley, -rrb-, who, hosted, the, infamous, red, wedding, during, which, her, brother, robb, stark, -lrb-, richard, madden, -rrb-, and, mother, catelynn, stark, -lrb-, michelle, fairley, -rrb-, were, killed, .

Figure 2: Tokenized form of sample text

4 Generative Language Models

In this project, three generative language models have been used. Each of these have been briefly explained in this section.

4.1 N-gram Markov Models

An N-gram model is a generative model that predicts the next word in a sequence based on the N-1 previous words, consistent with the Markov assumption. The probability of occurrence of a word then depends on the N-1 previous words only. In a unigram model, this corresponds simply to the probability of occurrence of a word in the corpus. In a bigram model, probability depends only on the previous word and so on. Mathematically, a bigram model can be represented as:

$$P(w_k) = P(w_k|w_{k-1}) = \frac{count(w_{k-1}w_k)}{count(w_{k-1})} \tag{1}$$

This is the earliest and perhaps the most commonly used type of generative models. However, these models do not inherently capture information such as the tense (past, present, future), voice(active, passive), grammar or context of the text. Also, higher order N-grams have a higher space and computation requirement and models with N greater than 5-10 are rarely used. This leads to the problem of short term dependency in the model which does not allow the model to prepare long sentences without them loosing coherence. In this project, simple bigram models with three different input training data sets have been explored and trigrams have been proposed in future work.

4.2 Hidden Markov Models

Hidden Markov Models are typically used for the purpose of part-of-speech tagging but being generative models in nature, they can also be used for the purpose of generating text. An HMM is a statistical markov model that assumes the output to be a markov chain governed by some hidden states. In our case of text generation, the hidden states have been taken to be the part-of-speech tags and outputs are words corresponding to each tag. In a first-order HMM, which has been used in this project, each PoS tag is generated using only the previous PoS tag in what is called a transition step and words corresponding to each tag are given as the visible output of the system in the emission step. Thus, a word only depends on its part-of-speech, which in turn only depends on the previous part-of-speech. As can be intuitively understood, HMMs would produce relatively grammatically accurate sentences since the PoS sequence is included in generating the sentence. However, since the word does not depend on the previous word, context will be lost completely and performance may even degrade as compared to simple N-grams. This has been found to be the case in some previous experiments and has also been corroborated in this project, where performance of HMMs has indeed been found to be poor.

4.3 Long-Short Term Memory recurrent neural networks

LSTMs [4] are a type of recurrent neural networks which were proposed in 1997 and have recently gained popularity as generative models in natural language processing. A recurrent neural network is a network which has internal states that are passed from one input to another, thus allowing it to exhibit a dynamic, temporal like behavior. The way LSTMs are utilized for sentence generation is similar to all other generative models i.e. they are trained with a corpus and provided with a seed, from which future words in case of a word-level LSTM or characters in case of a character-level LSTM are outputted. The unique thing about LSTMs however is that they can take in both a sequence or a single entity as input and provide a sequence or entity as output. In this project, both character and word level LSTMs have been used and trained over the entire corpus of fan theories with 20 character sequences as inputs and the next character as output. A 20 character sequence chosen randomly from the data set was provided as seed and a range of characters varying from 5 to 100 were taken as outputs.

5 Methodology

As described above, three generative models have been used to produce sentences from the training corpus. Along with the baseline model, variations in the training data to account for noise removal have also been performed in association with the N-gram models, resulting in two additional models thus making a total of five methods, which have been described below. Before performing language modeling, named entity recognition has been performed over the entire corpus and the count of each token recognized as a person has been taken to find the names that occur most commonly in the database. These names present the most famous characters and sample sentences have been generated only for these characters.

Method 1: Tokenized text + Bigram language model

The first method for generating text was a simple bigram model trained over the entire tokenized corpus. Tokenization was done as has been previously explained and dictionaries of bigram and unigram counts were respectively prepared. Once trained, seed word to the bigram was given as the name of a character from the list of the five most famous characters and 10 sentences for each character were generated using this model. For each word, the next word was chosen randomly provided it had a probability of occurrence above a certain threshold and the process was repeated till the period token '.' was sampled. Table 1 shows a seed word and a sample sentence generated for that seed using this method.

Table 1: Example sentences for five characters generated using method 1

Seed Word	Sentence
jon	jon and daenerys .
arya	arya fills the opportunity to some point next season .
cersei	cersei and jorah , thus refuses to reddit has to disagree and marry sansa certainly doesn 't craft steel .
dany	dany and go by ned stark ancestor .
ned	ned stark , which she told the popular fan , oh wait , aka merman .

In this method, no noise removal is performed and all the data is taken as it is. This results in the model learning from a lot of unwanted and redundant data which consequently affects its performance.

Method 2: OpenIE relation tuples + Bigram language model

As has been mentioned before, the Stanford OpenIE package, which is a part of Stanford CoreNLP[8] can be used during pre-processing to remove unwanted information or noise from a database. This is possible because the package extracts only relational tuples from the input text i.e. tuples in the form of subject-relation-object. Each sentence is first split into entailed clauses and each clause is then maximally shortened to produce shorter sentence fragments. These fragments are then segmented into triples and presented as outputs. This method performs noise removal to a certain extent as it ignores any information in a sentence that is not a part of the relation presented between the subject and object. This way, only the important information from any sentence is extracted and all additional information

is discarded. Using this method, 3500 relational tuples were extracted. Once the entire corpus has been passed through the OpenIE package and converted into relational tuples, another bigram model is prepared and trained over the new relatively noiseless corpus. Sentences are generated in the same manner as in method 1. Table 2 shows the seed word and sample sentence pair generated using this method.

Table 2: Example sentences for five characters generated using method 2

Seed Word	Sentence
jon	jon snow is faced god lol .
arya	arya take ned 's head is jon has sent as narcissist on daenerys goes .
cersei	cersei control .
dany	dany to greyscale .
ned	ned has pale .

One drawback of this technique of removing noise is that the same sentence is often shortened in multiple ways for multiple relations and thus, similar sentences are presented to the model repetitively. Also, certain relational tuples begin with a pronoun as their subject and provide no knowledge of who that pronoun might refer to, thus causing that tuple to lose any information that it may have otherwise provided.

Method 3: Named Entity relation tuples + Bigram language model

From method 2, we see that relational tuples extracted using OpenIE can be used to improve the quality of the corpus to be trained but this creates some new problems of its own. One such issue is the presence of a pronoun as the subject of a relational tuple. In this method, to further get rid of text that does not provide useful information, a list of names entities is prepared and from the extracted relational tuples, only the tuples with the subject matching an entity in the named entities list are kept while the others are discarded. This further narrows down the list of relational tuples from 3500 to 2000 but these tuples now rarely contain a sample which does not provide important information for the purpose of language modeling. An example of the type of sentences accepted and the type of sentences discarded has been given in table 3. Table 4 presents the seed words and sample sentences generated using this method.

Table 3: Examples of sentences accepted and rejected using named entity information

Accepted	Arya 's friend was seen in Season 3 Samwell Tarly Will Step Up to Plate
Rejected	his was temporarily freed citadel is with Gilly

Table 4: Example sentences for five characters generated using method 3

Seed Word	Sentence
jon	jon is true stark caused mad king 's son is research .
arya	arya arrived back trying .
cersei	cersei being character big enough .
dany	dany could threaten westeros love .
ned	ned stark heir .

Method 4: Tokenized text + Hidden Markov Model

Markov models, while good for generating sentences by learning form the sequential occurrence of words, cannot ensure grammatical correctness as they do not take into account the sequence of part of speech tags, which if in the incorrect order, present a grammatically incorrect sentence. Hidden Markov Models can be used for this purpose. Method 4 comprises of using a first order HMM with the hidden layer presenting the part-of-speech tag and the output layer presenting a word corresponding to each hidden node. Tokenized text has been used to train the HMM and given a seed word and its part of speech, the

remaining words have been determined. Sampling of words is again continued until the period token '.' is sampled. Table 5 shows the seed words and sample sentences generated using method 4.

Table 5: Example sentences for five characters generated using method 4

Seed Word	Sentence
jon	jon martin be an small states but the traditional theory kill was given her face looking things -rrb- , play faceless of if her burned knights have arya hound about we see not the winter
arya	arya littlefinger 's coat for it have n't , n't beloved brother under the walkers or the jorah load to get the mad stark , and somehow be
cersei	cersei euron m.o. 's downfall
dany	dany littlefinger and bran true that it how a secret rider on scenario , doubtful , gregor of nan cersei 's down so that thought he " dragons , structurally -rrb- and through dark scenario story and diagnose immediately in a jon troops
ned	ned bran as the idea

Method 5: Character level LSTM

In all the above approaches, only short-term dependencies have been taken into account. This means that occurrence of a word only depends on the previous word or its part of speech which in turn depends on the previous part of speech only. These short term dependencies cannot prepare good sentences, especially of a longer length since they will loose coherence as the length of the sentence is increased. Analysis of the sentences presented above shows that most sentences probably make sense in short fragments but do not present any overall fluency. This is due to the inability of the models to judge the next word based on a sequence of previous words. LSTMs have thus been used to include long-term dependencies when generating a sequence. Two types of LSTMs have been trained, once on the characters of the corpus, resulting into a character-level LSTM and once on the words of the corpus, resulting into a word-level LSTM. In both cases, a sequence (either characters or words) was taken as input and the next character/word was taken as output and LSTMs were trained with 2 layers, each consisting of 256 neurons. While testing, a randomly chosen sequence was given as input and a fixed length of words were generated for that seeding. Example of a character level LSTM seeded with 20 characters and the characters generated as output has been given in table 6.

Table 6: Example output of a character-level LSTM

arya starks kill li	st. in season 3 arya the the the the the

6 Evaluation and Results

Table 7 presents the top eight most popular characters of the TV show Game of Thrones and the number of times each name was encountered in the training corpus. Once these names have been determined, 10 sentences for each character have been generated using the five methods given above. For each sentence, two different ratings have been provided.

6.1 Evaluation Metrics

For the purpose of evaluation of the sentences generated using each of the aforementioned methods, human subjects were asked for assistance. 4 subjects, 2 of whom had domain knowledge while the other 2 did not, were asked to rate the sentences between 1 and 10. No fixed rules were provided to rank the sentences, in order to capture the general fluency and accuracy of each sentence along with how well a human reader can understand the sentence being generated. For reader with domain knowledge, this also captured how much information they believed to have received from a generated sentence and with respect to the task at hand i.e. generating fan theories, how important and meaningful they thought the sentence was. Based on ratings from these two groups of readers, two different metrics for rating

Table 7: Most famous character names and count of occurrence

Rank	Name	Count
1	jon	114
2	arya	62
3	cersei	55
4	dany	44
5	ned	32
6	ramsay	31
7	jaime	28
8	sam	27

sentences were considered. Metric 1 of evaluation was general fluency and clarity in meaning of sentences without taking domain knowledge into consideration. Metric 2 of evaluation was fluency, accuracy and information content of sentences taking domin knowledge into consideration.

The two metrics of evaluation were taken in order to capture how well each method performs in general versus how meaningful its outputs are when thought of in terms of a particular domain. Each method was assessed independently on the basis of how well it could capture domain knowledge and different methods were compared with each other to see which gave the best performance in each of the two areas.

6.2 Results

Comparison of the first four methods using metric 1 i.e. without taking any domain knowledge into consideration has been shown in figure 3. Ratings have been calculated by taking the average of the rating of all 50 sentences corresponding to each of the models. This result shows that method 3 performs better than all the other methods when only general coherence is considered. This can be attributed to the fact that only important information and sentences which are properly formed, with a proper noun as subject and a relation between subject and object, were the only ones given to the model for training. This resulted in a more robust trained bigram model which in turn resulted in better sentences. HMMs performed quite poorly as compared to even the baseline bigram model. This can be justified by shedding light on the fact that while the grammatical structure of the sentences would be intact owing the the appropriate part-of-speech sequence, words corresponding to each PoS tag are generated irrespective of what the previous word might be. This results in sentences which do not have much continuity as the words, despite having the correct grammatical ordering, are still not related to one another, resulting in an informative sequence.

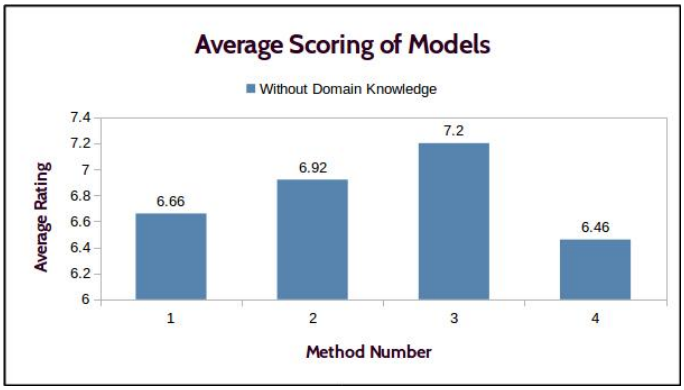

Figure 3: Comparison of methods using metric 1

Figure 4 shows a comparison between the methods based on ratings when domain knowledge is also taken into account. As can be seen, HMMs still perform quite poorly and are no better than the baseline bigram model. However, in this situation, the second method, which still keeps all relational tuples intact, seems to perform the best, as opposed to method three in the previous case. This can be reasoned in the following manner: since the second method keeps all the relational tuples while removing noise from the database, it performs better than method 1. However, method three further reduces the number of tuples and while this provides a generally better formation of the resultant sentences, loss of information of domain in reducing the tuples from 3500 to 2000 results in an overall poor performance of that method domain-wise. Because the second method still has sufficient sentences to learn some information about

the domain while taking care of all the noise, it performs the best when sentences are evaluated keeping the domain information in mind.

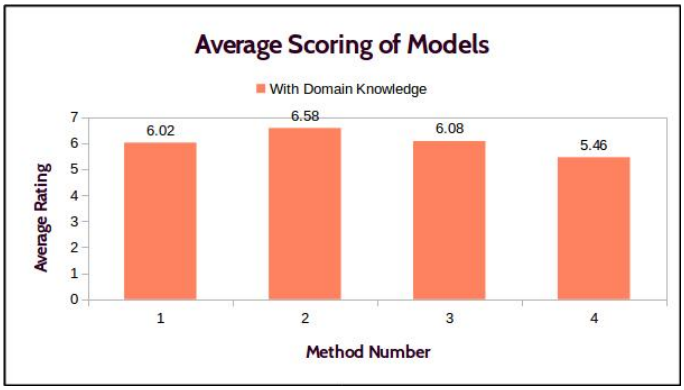

Figure 4: Comparison of methods using metric 1

The final result has been presented in figure 5, which shows, for each of the 50 sentences, the ratings given with and without domain consideration. As can be seen, in general, rating of a sentence without taking domain knowledge into account is higher than when domain knowledge is considered. This pattern holds true for all four models, where for each one, average ratings with domain knowledge are lower than average ratings without domain knowledge. This can again be attributed to the fact that while a sentence may in general make sense to common audience, when taken the perspective of domain-based information, the sentence may not be considered that important or informational.

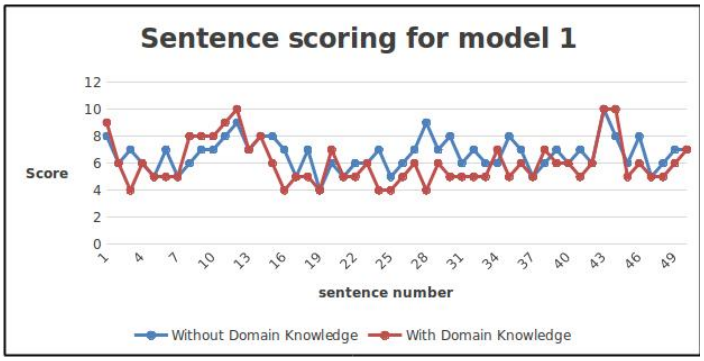

Figure 5: Comparison of ratings using metrics 1 and 2 for method 1

From table 6 it can be seen that after a certain number of sampling epochs, the model starts repeating. This is probably due to the fact that words such as articles, punctuations etc are more commonly occurring that other words, which is why when selecting the most probable next character, the LSTM keeps choosing the same most common character again and again. This results in the formation of a loop where the same character sequence keeps repeating. The same happens in a word-level LSTM as well.

7 Conclusion

In this project, sentence generation methodologies for the case of fan theories of the TV show Game of Thrones have been explored. The database has high noise, contains jargon and is sparse in nature. The performance of different generative models on such a database have been explored and standard NLP packages such as Stanford CoreNLP and Stanford OpenIE have been exploited for the purpose of improving the performance of one of these models. It has been found that simple generative models such as bigrams in general perform better that the more sophisticated HMM and LSTM models for a noisy database such as this one. Also, it can be concluded from figure 5 that these generative models are not good in learning domain information from a training corpus and in general do not map information relevant to the particular domain in their generated results. From the performance of bigram models after noise removal, it can be concluded that use of relational tuples for this purpose can indeed improve the performance of standard bigrams, which also leads to the conclusion that input training data is an

important factor when preparing a model and it is a decisive factor in determining how accurately the model performs. In general, the Stanford CoreNLP and OpenIE packages perform well, with appropriate PoS tagging, Named Entity Recognition and Co-reference resolution.

8 Discussion and Future Work

It is interesting to see that despite the superiority of recurrent neural networks, they perform very poorly in the presence of a noisy database, while models as basic as bigrams give a better than average performance. Right now, only bigram models have been used and experimentation with some noise removal techniques have been conducted. However, there are multiple ways in which this project can be carried forward. Some of the possibilities have been discussed here.

1. **Higher N-grams:** Despite the fact that higher degree N-gram models are rarely used, models such as trigrams or quadruple grams can still be used. Intuitively, these models combined with the noise removal techniques explored above should provide readable and well formed sentences.

2. **Co-reference resolution:** Instead of removing tuples having a pronoun subject, co-reference resolution can provide us with the information of which noun that pronoun points to. If each pronoun is replaced with its corresponding noun and then tuples beginning with named entities are extracted, the size of database would be comparatively bigger and more information will be passed to the models. This should ideally result in improvement of the domain rating as well as the general rating of the models.

3. **Relational Tuples and LSTMs:** The reason concluded for why LSTMs perform poorly is that presence of words such as 'the','a','for' etc overshadows the other informative content. In that case, training the LSTM with relational tuples extracted using OpenIE instead of the entire corpus can help solve this particular problem.

4. **Word embedding with LSTM:** Word-level LSTM in this project was trained by indexing each word and considering the word index to be a feature of the word. However, more descriptive features such as word embeddings can be used to train the LSTMs in order to improve their performance.

These methods mentioned above could not be attempted due to lack of time and/or lack of computation space and power. However, with sufficient computation space, powerful LSTMs can be trained to learn long-term dependencies and these, trained over the noiseless relational tuples with word-embeddings as features can be expected to outperform the current models by a significant margin.

Database Sources

For the purpose of database collection, several websites were scanned and theories were hand-picked and stored in text files. The following websites are some of the main sources of the training corpus collected for this project.

Game Of Thrones - A REDDIT OF ICE AND FIRE
Mashable: 10 GoT Fan Theories Ranked Least to Most Likely
Mashable: 9 key predictions for what'll happen in 'Game of Thrones' Season 7
Mashable: Here's what 'Game of Thrones' super-fans think will happen in Season 7
Cosmopolitan: 17 Most Insane Game of Thrones Fan Theories
Game-of-Throne-best-fan-theories-season-6-spoilers
Game-of-Thrones-wildest-fan-theories
Game-of-Thrones-theories
17-big-Game-of-Thrones-fan-theories
The-13-craziest-Game-of-Thrones-fan-theories
Game-of-Thrones-8-more-fan-theories-for-season-7-and-8
Grading-the-latest-Game-of-Thrones-fan-theories
15-amazing-Game-of-Thrones-fan-theories
'Ggame-of-Thrones' season-7-theories

References

[1] Barbara Gawronska and David House. Information extraction and text generation of news reports for a swedish-english bilingual spoken dialogue, 1998.

[2] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[3] Brian Hieu Nguyen. Rap lyric generator, 2009.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[5] Mizuki Kobayash, Ichiro Kobayashi, Hideki Asoh, and Sergio Guadarrama. A probabilistic approach to text generation of human motions extracted from kinect videos, 2013.

[6] Irene Langkilde. Forest-based statistical sentence generation. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 170–177, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[7] Mirella Lapata. Probabilistic text structuring: Experiments with sentence ordering. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 545–552, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[8] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

[9] Ehud Reiter, Roma Robertson, and Liesl M. Osman. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*, 144(12):41 – 58, 2003.

[10] Grzegorz Szymanski and Zygmunt Ciota. Hidden markov models suitable for text generation.

[11] Yezhou Yang, Ching Lik Teo, Hal Daumé, III, and Yiannis Aloimonos. Corpus-guided sentence generation of natural images. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 444–454, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[12] David Zajic, Bonnie Dorr, and Richard Schwartz. Automatic headline generation for newspaper stories. In *IN THE PROCEEDINGS OF THE ACL WORKSHOP ON AUTOMATIC SUMMARIZATION/DOCUMENT UNDERSTANDING CONFERENCE (DUC*, pages 78–85, 2002.