FRI Natural language processing course 2023
Project report

University of Ljubljana
Faculty of Computer and Information Science

Literacy situation models knowledge base creation

Timotej Košir, Luka Brecelj, Matjaž Ciglič

Abstract

This article addresses the challenge of generating story endings with context using text generation models. We propose a model that generates story endings based on a story without a conclusion, taking into account the given context in the form of character sentiment. The approach involves two domains: character sentiment analysis and story generation. The article discusses related work in automatic story generation, including approaches using spatiotemporal modeling, controllable multi-character interactions, and large-scale latent variable models. We describe the dataset used, which consists of fairy tales from around the world, and the challenges associated with the dataset's inconsistencies and unmarked story endings. We present a methodology for sentiment analysis of characters, story generation using a fine-tuned GPT-2 model, and evaluation techniques including perplexity scores and sentiment analysis comparison. The article concludes with preliminary results and future directions for fine-tuning the model.

Keywords

NLP, GPT-2, story generation, ending, sentiment analysis

Advisors: Aleš Žagar and Slavko Žitnik

Introduction

In recent years, there have been significant advancements in text generation models, but they still face challenges in producing coherent and lengthy stories with context provided as additional knowledge. Our paper addresses this issue by developing a model that can generate ending based on a story without a conclusion, while taking into account given context. Context will be provided in form of character sentiment, which will be reversed. To achieve this, we divide the problem into two domains: character sentiment analysis and story generation. With the former, we aim to extract information about the attributes of individual characters in the story, while the latter focuses on using the provided data as additional context for story generation.

Related work

Article Automatic Story Generation: Challenges and Attempts [1], provides a good baseline. It describes an approach to spatiotemporal modeling using convolutional LSTM networks (CLSTM). The authors introduce a Bayesian hierarchical framework that employs low-rank matrix factorization to learn separable spatial and temporal priors for the CLSTM weights [1]. Furthermore, they incorporate uncertainty quantification into the framework using dropout variational inference. Experiments on synthetic and real-world datasets demonstrate the efficency of the proposed method, with improvements in predictive performance and uncertainty quantification compared to existing techniques [1].

More similar to our area is an interesting article titled Controllable Multi-Character Psychology-Oriented Story Generation [2]. It presents an approach to automatic story generation that emphasizes controllable multi-character interactions and psychological elements [2]. The authors propose a psychologically-driven, deep-learning-based framework that uses BERT-based models for character interaction, an LSTM model for plot control, and a psychology-based rule system to manage character relationships and emotions. Experiments demonstrate the effectiveness of the proposed method in generating coherent and engaging stories with customizable plots, complex character relationships, and a rich portrayal of character psychology, highlighting its potential for advancing the field of story generation.

In the article[3], authors developed a text generation framework to address the disadvantages of previous model[4]. They incorporated external knowledge and developed a contextual ranker to rank the relevance of retrieved knowledge sentences to the story context. Sentence embedding is then used for weak supervision. The top-ranked knowledge sentences are fed to the conditional text generator to guide generation. The results are measured by automatic metrics and human evaluations, which show that their model generates more fluent, consistent, and coherent stories with lower repetition rate and higher diversities compared to previous state-of-the-art models. The model is also controllable as evidenced by successful replacement of keywords used to generate stories. Especially beneficial for us are the comparison between BERT and GPT-2 languange models and ways how to compare goodness of generated stories.

Very recent article on the topic[5] explores the use of large-scale latent variable models for neural story generation with a focus on controllability. Authors integrate latent representation vectors with a Transformer-based pre-trained architecture to build a conditional variational autoencoder (CVAE) and show through experiments that their model achieves state-of-the-art conditional generation ability while maintaining excellent representation learning capability and controllability. The model components, such as encoder, decoder, and variational posterior, are all built on top of pre-trained GPT-2 language model.

Dataset

We opted for the "Fairy Tales from Around the World" dataset [6] due to its rich and diverse collection of 1076 stories. The stories vary in length from 176 to 31420 words, with an average of 2370.1 words. They come from various countries and traditions, providing a more suitable foundation for our project. However, inconsistencies in the stories and unmarked story endings still need to be addressed.

To tackle these issues, we devise a system that approximates each story's ending by considering the final 10% of the text. This method enables us to generate an initial dataset with rough approximations of the story endings. Although not flawless, it served as a valuable starting point for further refinement. We opted for a combination of computational and manual efforts, acknowledging that the automated procedure might not yield a perfect dataset.

After the initial processing of the dataset, we proceeded to verify whether each file contained a complete story from start to finish, as well as the correct placement of the special character $< EOS >$ between the story and the ending. Additionally, we removed quotation marks and replaced exclamation points and question marks with commas within each story. This step allowed us to eliminate stories that were presented unconventionally and had ambiguous or incomplete endings. This process helped maintain the dataset's integrity and quality.

Since we are dealing with stories, we have to take into account complex structure that our dataset will have. Stories often contain intricate, multi-layered plots with rich character development and various themes, which proved to be a challenge for the GPT-2 model. The model struggles to capture and understand these complexities, resulting in poor prediction performance.

Another challenge arose from the length of the stories. The dataset consists of stories, varying in length, from 176 to 31,420 words. This significant variation in story length made it difficult for the model to form a consistent learning pattern. Longer stories contain more information, characters, and plot developments, making them harder for the model to grasp and learn effectively. The GPT-2 model was originally trained on a vast range of internet text. While this enables the model to generate diverse and creative text, it does not guarantee that the model will perform well on specific, highly-structured tasks like understanding and generating a fairytale.

Given these difficulties, we turned our attention to the ROC Stories Corpus from 2018 [7]. We used our fine-tunned GPT-2 as a better starting point and trained it on the ROC Stories Corpus. Available stories were separated into a training set and an evaluation set. The training set consists of 54,235 stories, while the evaluation set contains 1,572 stories. The stories in the ROCStories Corpus are simple, short, have uniform and straightforward narrative structure, making them suitable for relatively small GPT-2 model.

Personality traits dataset

Pretrained transformers for sentiment analysis are usually trained on twitter posts, movie reviews, ... which are not suitable for character sentiment prediction. One good example of that would be distilbert-base-uncased-finetuned-sst-2-english, used as a default sentiment analysis transformer in Hugging Face pipeline. If we give it sentence "Bob is a murderer." we will obtain negative sentiment. But if we provide the prompt "Bob is a murderer, but in love with Annie." the sentiment becomes positive. Because of such inaccuracities, we decide to fine-tune the model. After searching the internet we did not find a suitable database, which would be explicitilly created for personality traits, so we created our own.

Maual labelling of the data is not viable due to time constraints, so we resort to text mining approach. Firstly we collect a list of different personality traits from 9 different online sources, with most prominent being Character traits list [8], 1000 NPC traits [9] and 500 Words Characterization Reference Support Tool [10]. In the next step we find list of common words with their positivity and negativity scores, AFINN [11], Sentiment Lexicon [12] and Subjectivity Lexicon [13]. Each unique word from the sources is assigned a score, which will range from -5 to 5. Negative numbers represent negative words, 0 neutral and positive number positive sentiment. We use these scores to tag our character trait words. In total we had to label 212 traits by hand.

In the end we are left of with 827 positive, 1015 negative and 72 neutral traits. Obtained character trait sentiments help us to automatically label the dataset. For that we take our fairytales dataset, split it into sentences and search for any mention of a trait, which would indicate the character sentiment. Inititally, this labels us 12498 out of 119223 sentences. To label even more data, we add nouns that indicate person's sentiment such as murderer, thief. In the end we automatically label 30 % of the sentences.

	C1	C2	C3	C4	C5
1	David noticed he had put on a lot of weight r…	He examined his habits to try and figu…	He realized he'd been eating too much fa…	He stopped going to burger places and st…	After a few weeks, he started to feel much b…
2	Tom had a very short temper.	One day a guest made him very angry.	He punched a hole in the wall of his hou…	Tom's guest became afraid and left quick…	Tom sat on his couch filled with regret abou…
3	Marcus needed clothing for a business casual …	All of his clothes were either too for…	He decided to buy a pair of khakis.	The pair he bought fit him perfectly.	Marcus was happy to have the right clothes f…
4	Bobby thought Bill should buy a trailer and h…	Bill thought a truck would be better f…	Bobby pointed out two vehicles were much…	Bill was set in his ways with convention…	He ended up buying the truck he wanted despi…
5	John was a pastor with a very bad memory.	He tried to memorize his sermons many …	He decided to learn to sing to overcome …	He then made all his sermons into music …	His congregation was delighted and so was he.
6	Melody's parents surprised her with a trip to…	Melody took a nap during the two hour …	When they arrived, Melody was energetic …	At the aquarium Melody saw sharks, tropi…	After five hours at the aquarium, Melody and…
7	The math teacher announced a pop quiz as clas…	While some students complained, he beg…	I took out my pencil and began to work.	About 5 minutes later, I finished.	I stood up feeling confident and turned it i…
8	My first girlfriend i met on the internet.	She lives about 4 hours away from me.	Finally after 2 years we met each other.	She stayed with me for a week or two.	We decided we couldn't be apart so she moved…
9	I got Charlie Horse when I was four years old.	He's a brown stuffed horse, and at 35 …	He was my best friend, and always laid a…	I laid him next to me, smelling his soft…	I liked to listen to my radio as I fell asle…
10	Laura loved corn.	So she decided to grow some in her bac…	The whole process of growing them made h…	But she realized that they required too …	So Laura quickly abandoned her corn garden i…
11	Andy was invited to a Halloween party.	Andy figured that for dramatic effect,…	Since Andy's costume was green, Andy dec…	After the stylist finished the coloring,…	Andy was disappointed with his new, bold, gr…
12	Luke was playing hockey at school.	The game was tied and almost over.	Then Luke made the winning shot!	Everybody cheered!	Luke was so proud of himself!
13	Robbie was competing in a cross country meet.	He was halfway through when his leg cramped up.	wasn't sure he could go on.	He stopped for a minute and stretched hi…	Robbie began to run again and finished the r…
14	Jude was very excited about his college gradu…	On the way to the arena, he got stuck …	He only had an hour before the ceremony …	He thought he wasn't going to be able to…	Luckily, the traffic cleared up in time for …
15	Beth sent a letter to Santa Claus.	She received a letter back in the mail.	Beth did not think that it sounded like …	She sent another letter calling that San…	She did not receive another letter back in t…
16	Our granddaughter Anna is very fussy about he…	She is only two, but wants to pick her…	Today her mom wanted Anna to wear a dres…	Anna started crying.	They let her wear a t-shirt and shorts inste…
17	Jake needed a ride to the store.	His girlfriend was working and wouldn'…	His brother was in the middle of a movie…	Jake decided to take the bus to the store.	He got what he needed all by himself.
18	I'd been looking for a stand for my TV.	I looked everywhere, but the ones I sa…	I happened upon a yard sale on my way ho…	She had a stand that was perfect, and ch…	It looks great in my den.
19	Sally had a root canal this morning, as she h…	After the procedure, the dentist wrote…	She headed straight to the pharmacy to f…	She handed the prescription to the techn…	The technician called her name and she paid …
20	My dog is terrified of thunder.	There was a storm today.	He came running into my office.	He hid in the kneehole of my desk, tremb…	Once the storm was over, he acted like a bra…
21	Mark and Jo were brainstorming ideas for a ch…	Mark suggested that a monster attacks …	Jo laughed because he thought Mark was j…	Mark was confused because he thought it …	Mark left in a huff for having his ideas moc…

Figure 1. Part of ROC Stories Corpus 2018

Methods

This section presents the methodology needed to perform various tasks in our pipeline. Initially, we obtained the character sentiment and combined it with the dataset. By combining both the story and context, we obtained prompts that were used to train our story ending generation model. We will describe the model and explain the necessary steps to ensure it's good performance and how we evaluate it.

Sentiment Analysis

To obtain sentiment analysis of the characters, appearing in a given fairytale, we follow three main steps: character detection, extraction of sentences containing character mentions, and sentiment analysis of the extracted sentences. The implementation uses Python and utilizes libraries such as spaCy, NLTK, and the Hugging Face Transformers library.

To detect characters in each story, we employ spaCy's pre-trained en_core_web_trf model to perform named entity recognition. We focus on entities labeled as "PERSON" and add them to a set, ensuring unique character names. Then, we tokenize the story into sentences and search for each character's name in the sentences using regular expression searches. Once a match is discovered, we pull out a series of sentences that include the reference to the character. After collecting all the sentences for each character, we utilize our fine-tuned sentiment analysis transformer to obtain a score for each sentence. These scores are averaged to produce the final score for the character's sentiment.

The detected characters and their sentiment scores are stored in a JSON format, with one entry for each story. The resulting JSON file, "dataset_sentiment.json", contains a representation of the characters and their associated sentiment scores. The entire process is executed for every story in the dataset.

Story Generation

In order to exploit the power of pre-trained models, we use the smallest version of GPT-2 model [14] for story ending generation. It has 12 layers, 12 heads per layer, input size of 1024 units and total of 117M parameters. For our intended use, it's important to have an encoder that features an unmasked or bi-directional structure, which provides a more comprehensive information than the masked or uni-directional structure typically found in the decoder for auto-regressive generation. Since GPT-2 only accepts a prompt as input to initiate text generation, we need to embed the necessary information about the desired context for story generation into the prompt itself. We accomplish this by defining a special token, $< sep >$, which separates the text we want to continue from the context. The prompt takes the form of "context + $< sep >$ + sentence", which is then encoded into the latent space using the GPT-2 tokenizer.

To fine-tune the model, we use the dataset of stories with obtained character sentiments. Ideally, we would use the whole story up until the end and all the character names with their sentiments reversed as the prompt. But GPT-2 model accepts at most 1024 tokens as input, so we would need to have some internal state that would remember the past inputs of the story or embed the whole story into limited amount of tokens. To solve the problem, we currently split the story into windows of 500 words, which gives us about 500 tokens for each window to train the model on. Another problem is also too broad character sentiment score, which is a decimal in range from -1 to 1. We limit it by constraining the score to arbitrary amount of groups of positive, neutral and negative sentiments. Due to limitations of the sentiment model there will only be negative or positive sentiment value.

Evaluation

After fine-tunning the GPT-2, we have to evaluate the generated text. Since there are no target texts we can compare our result to, we can use combination of automatic metrics and human evaluation.

One of such metrics is perplexity. Commonly used in the context of language modeling, it measures how well a model can predict a sequence of words by assigning a probability score to each possible word, based on the previous words in the sequence. The lower the perplexity score, the better the language model is at generating text. In our case, we split the test dataset into sentences. Each word is then a prompt for the language model to generate a new story with a given context. We thus calculate the perplexity score of the generated ending given the context.

Comparing the sentiment of characters in the original and generated story is another approach to automatically evaluate the performance. In this approach, the sentiment of each character is analyzed to determine whether the sentiment in the story aligns with the opposite character sentiment in the generated text. For example, if the writing prompt requires a story with positive character sentiment, then the sentiment analysis of the generated story's characters should reflect the opposite.

Last proposed approach is manual comparison of the generated story ending with the given context. After generating several stories, a human evaluator can manually compare the coherence, relevance, and alignment of the story with the context. The best story out of them is chosen and we fine-tune the model accordingly. Specifically, if the story is an existing one, the ending can be added to dataset and used as a target to teach the model. Contrary, if the story is unseen, it can be added as a new example to our dataset.

Results

First step was to perform sentiment analysis of the characters. It is not a focus of our project, so we did not dive too deep into it. To detect characters, we used NER [15][16] model. Since it was not trained to detect people from stories, some non-named entites, such as witch, mother or brother get lost. But overall, only 43 stories did not have any detected characters. These stories were read and we extracted the characters manually. Additional issue when extracting character sentiment arises with the use of pronouns (e.g., he/she), since the model does not recognize them as referring to a specific name. To address this challenge, we have expanded the context to include more sentences after a name is mentioned. Each sentiment is calculated as average of all the sentences around where the character appears. Every sentence gets score calculated by our sentiment model, which was trained for 5 epochs with batch size of 64 from our Sentiment Analysis Dataset. Empirical analysis shows, that overall the obtained sentiments are more accurate than they were before.

To fine-tune our story generation GPT-2 model we used Google Colab, utilizing an A100 40GB GPU to meet the considerable computational demands of the training. Even the smallest version of GPT-2 is resource-intensive. Several experiments were conducted to identify the ideal configuration for our task, such as different batch sizes, amount of batches per update and amount of tokens per window. We found out that 10 epoch with batch size of 8 yielded the best results. One of the significant challenges we faced was the addition of special tokens, which confused the model weights, causing GPT-2 to generate tokens instead of coherent texts. We managed to address this issue when the amount of epochs was increased.

The results of the generated stories are not promising. Model is able to generate somewhat coherenent continuation of the story, but fails to make a relevant connection to characters in the story. The model does not take context into account and often weers off on a tangent, which does not provide a satisfactory ending. A particularly common issue we also noticed was the model's tendency to hallucinate characters. It often introduced popular characters from other, unrelated sources, like Harry Potter, despite such characters never appearing in the training data. These hallucinated characters further confused the story structure and detracted from the original narrative.

For the first evaluation approach, we have implemented sentiment comparisons. Each story has two original endings and 5 generated endings with negated context. The sentiment was checked for each character in the story in both the original endings and the generated endings. For the 40 input stories, that resulted in 420 sentiment comparisons. 178 comparisons resulted in correctly generated endings, which represents 42.38% of the comparisons. However, most of these endings were classified as "neutral", most likely due to the use of pronouns and other words the model failed to detect and thus returned "neutral" results. In fact, only 10 comparisons yielded expected results, which is only 2.38% of all comparisons or 5.62% succesful comparisons.

We have also implemented the perplexity metric for evaluation, which we thresholded with different threshold values. The results are displayed in table 1.

Threshold limit	Number of PBT	Percentage of PBT
3	0	0%
5	7	3.5%
7	19	9.5%
10	38	19%

Table 1. Number and percentage of perplexities below the threshold (PBT), out of 200 sampled.

We expected the lower perplexities would correlate with the succesful sentiment comparisons, but the results did not display any notable correlation. In fact, some of the perplexities, that achieved good results were in range $24-27$, which is among the higher results we obtained.

However, our results are similar to the ones obtained in article [5]. The model used in that article does use longer stories, whereas ours uses only stories of only a few sentences. And since the citated article uses a larger model with multiple layers, our approach had a somewhat lower performance.

Alongside our efforts with GPT-2, we also turned to the larger and more powerful model available via OpenAI's ChatGPT API, specifically GPT-3.5-turbo. We instructed this model to generate the story ending while taking into account sentiment analysis. Notably, these results were achieved without any fine-tuning of the model on our specific task. We issued subsequent prompts to GPT-3.5-turbo, providing only the story without the ending, and the sentiment of the character. Despite the absence of an explicit ending and without task-specific fine-tuning, the model performed impressively. This outstanding performance is likely attributable to the sheer scale of GPT-3.5-turbo. To be precise, it has 154 billion parameters, a significant increase compared to the 117 million parameters of GPT-2.

```
PROMPT: Your task is to generate story ending where you also take into account
sentiment. For example here you have story without ending: At school, Mary received
 an assignment to write an essay about pandas. Luckily, Mary loved pandas, as was
excited to get started. As soon as she got home, Mary booted up her computer and
began. She wrote about all she knew and researched a little too! Now you generate
story ending (1 sentence), where Mary sentiment is negative and correct story
ending is: Mary disliked Pandas.
OUTPUT: As Mary neared the end of her essay, she couldn't help but feel a tinge of
disappointment, realizing that her extensive research had only deepened her
distaste for pandas.
PROMPT: Now write me story ending for this story where Anna sentiment is positive:
Anna went to the salon. She was getting her nails done for the first time. She was
nervous, but the tech was friendly. He did her nails quickly and expertly.
OUTPUT: As Anna admired her beautifully manicured nails, a wide smile spread across
 her face, filled with joy and excitement at the newfound beauty and confidence
they brought her.
```

Figure 2. GPT 3.5 Turbo: Example of using GPT-3.5-turbo for generatin ending of story by considering the sentiment.

Discussion

Our approach presents a good starting point to the topic of the project. But there are many improvements to be made. One of the biggest issues in named entity recognition are non-named entities (brother, mother, witch, ...) and pronouns. To combat this, we could further fine-tune the NER model to detect such entities and use Coreference Resolution model [17] to replace the pronouns with corresponding names.

Another possibilty of improvement is sentiment detection. Since sentences in stories are convoluted and there is no repetative structure, we need a bigger dataset. Presented approach improves the results even though it uses score labels from traditional text mining techniques. Reason is that transformers take into account contexts of the words, not just the scored word. To further train the model, we could use self-supervised learning. If we manually label the most difficult sentences, the model will improve over iterations. Transfomer currently only has ”positive” and ”negative” sentiment, which should be expanded to more different values. On the other hand, we could use a completely different approach and train the model directly to detect entites from the stories and their sentiment, but it would require a complete restructuring of our story generation pipeline.

Considering that LMM such as GPT 3.5, work well when asked to extract sentiment and finish the story with a desired ending, we assume the problem of GPT-2's poor performance lies in the model being too shallow. We could use a larger model, such as GPT-2 Large or mpt-7b-storywriter-4bit-128g [18], which accepts 63k tokens and would be able to fit the whole story in a single input. But in practice, such approach is not viable, since the fine-tunning requires huge amount of data and resources. The performance of GPT-2 could also be improved by self-supervised learning as proposed, but we did not have enough time to have a human evaluator.

Given the rich character development and intricate plots in the Fairy Tales from Around the World dataset, we experienced difficulties when fine-tuning the GPT-2 model, which demonstrated poor prediction performance due to the complexity of the data. The substantial variation in story length added another level of difficulty, as the model struggled to form a consistent learning pattern. To address these issues, we utilized the ROC Stories Corpus from 2018 [7] to further train our GPT-2 model. This corpus, with its simpler, shorter and more uniform narratives, provided a more manageable task for our model.

Conclusion

In conclusion, the generation of story endings based on incomplete stories is a challenging endeavor. Our study presents a model that combines character sentiment analysis and story generation to produce coherent and contextually appropriate conclusions. This model was designed to tackle the complexities of varying narrative structures and the diverse range of story lengths found in the Fairy Tales from Around the World dataset [6].

The evaluation of the generated story endings was done using a combination of sentiment analysis and perplexity scores. Despite the results falling below our expectations, the work we have done provides a foundation for further advancements in the field. This process highlights the potential for improving the capabilities of text generation models, allowing them to create endings that align more accurately with the desired context and provide satisfying conclusions to incomplete stories.

References

[1] Amal Alabdulkarim, Siyan Li, and Xiangyu Peng. Automatic story generation: Challenges and attempts. *arXiv preprint arXiv:2102.12634*, 2021.

[2] Feifei Xu, Xinpeng Wang, Yunpu Ma, Volker Tresp, Yuyi Wang, Shanlin Zhou, and Haizhou Du. Controllable multi-character psychology-oriented story generation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1675–1684, 2020.

[3] Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Anima Anandkumar, and Bryan Catanzaro. Megatron-cntrl: Controllable story generation with external knowledge using large-scale language models. *arXiv preprint arXiv:2010.00840*, 2020.

[4] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.

[5] Le Fang, Tao Zeng, Chaochun Liu, Liefeng Bo, Wen Dong, and Changyou Chen. Transformer-based conditional variational autoencoder for controllable story generation. *arXiv preprint arXiv:2101.00828*, 2021.

[6] Anna Bengardt. Fairy tales from around the world, 2021. Kaggle Dataset.

[7] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*, 2016.

[8] Character traits list. https://www.chompingatthelit.com/character-traits-list/. Accessed: [Date].

[9] 1000 npc traits. https://www.roleplayingtips.com/tools/1000-npc-traits/. Accessed: [Date].

[10] List of character traits - 500 words characterization reference support tool. https://www.teacherspayteachers.com/Product/List-of-Character-Traits-500-Words-Characterization-Reference-Support-Tool-7431769. Accessed: [Date].

[11] Finn Årup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, 2011.

[12] Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[13] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Opinionfinder: A system for subjectivity analysis. *Computational Linguistics*, 30(1):25–46, 2005. Subjectivity Lexicon available for separate download.

[14] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[16] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[17] Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. *arXiv preprint arXiv:1804.05392*, 2018.

[18] MosaicML NLP Team. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. Accessed: 2023-03-28.