Smart Vet: Autocompleting Sentences in Veterinary Medical Records

Samuel Ginn
Department of Computer Science
Stanford Unviersity
Stanford, CA 94301
samginn@stanford.edu

Abstract

Every day, veterinarians write tens of thousands of medical records, mostly in standard formats following the SOAP structure: "Subjective", "Objective", "Assessment", and "Plan". These notes record the findings of their physical exams and observations of their patients, and take countless hours to write. We present in this paper a new system that we call "Smart Vet" that assists veterinarians in the writing of their notes by suggesting autocompletions for their sentences as they are writing them within the sections of their medical records. To enable this, we present two approaches: an end-to-end deep learning system that models this task as a seq2seq neural machine translation problem (i.e. translate a given sequence of sentences that correspond to the existing medical record into the following sequence that corresponds to the next sentence the veterinarian would want to write) and a transformer-based language modeling system based on OpenAI's recent advancements. Based on the success of this latter method, we evaluate this system live in a medical records application, and successfully see our autocompletions being used in production 12.46% of the time—a remarkable success.

1 Introduction

In 2016, the American Medical Association sponsored a study to investigate the amount of time that doctors spend writing their medical records. The study found that for every hour that a doctor spent face-to-face with a patient, they spent an additional two hours on their electronic health records (EHR) system writing up their notes [1]. This was in addition to the approximately 86-minutes they spend at home after hours finishing their notes in what has become commonly known as "pajama time" [2]. Doctors, trained for years to be medical professionals, are having two-thirds of their time every day relegated to glorified data entry. Forced to spend an absurd amount of time typing on the computer, doctors everywhere are becoming more and more frustrated with their technology and demanding change. Understandably, tedious EHR use is one of the leading causes for physician burnout [3]. The time-consuming nature of documenting patient visits does not only harm physicians' well-being, but it also hinders the patient experience: patient satisfaction is hurt with the adoption of EHRs due to the increased amount of time that doctors spend writing on their computers rather than conversing with patients (even in the exam room) [6]. It would bring about untold amount of good if one could reduce the amount of time that doctors need to spend on this necessary burden.

In veterinary medicine, although no rigorous study has been conducted, the numbers are probably far worse, because, although they have the same documentation requirements as human doctors, they have far less support staff and far worse electronic interfaces. Yet, many of the notes that these veterinarians and doctors are writing are repetitive in nature, as they see similar cases over and over. If we were able to speed up the process in which veterinarians or doctors would be able to write their notes, that would result in dramatic improvements: increased job satisfaction, more face-to-face

interaction with patients, and, perhaps, even the ability to see more patients in a given day. This paper hopes to provide a first-step to enabling this by providing autocompletion suggestions for veterinarians while they are writing their notes.

Due to the severity of the problem, many solutions have been attempted to help negate the issues doctors and veterinarians face. Some hospital groups pay for full-time "scribes" or medical assistants whose sole job is to follow the doctor around during the day and write their notes for them—an expensive position. Others have outsourced the "scribe" process by providing virtual scribes that watch the clinician's day through wearables, like Google Glass (see Augmedix's commercial product). Many, including some of the top technology companies, are now exploring ways to leverage artificial intelligence to make a difference, including trying to build completely voice-based AI scribes. But, none of these pursuits have turned out to be cost efficient or successful yet (hiring a full-time scribe, overseas or in person is prohibitively expensive at scale). Regardless, the amount of resources being deployed to try to solve this problem is further evidence of the importance of the task at hand.

Both veterinarians and human doctors record their notes of every patient visit and exam in what are called SOAP notes. These notes consist of Subjective, Objective, Assessment, and Plan sections. Subjective records the history of the patient, objective delineates the status of the physical systems, assessment records the diagnosis, and the plan section prescribes the ordered treatments. The composition of a SOAP note is called an "encounter." Our goal with this research is to provide a way to alleviate the process of writing these SOAP notes using live AI autocompletion.

2 Related Work

There have been several attempts in recent years to help alleviate doctors' problems with artificial intelligence that can learn to model medical language. We highlight some recent efforts that have inspired this work as well as models in other domains that have seen real success in the field.

In 2018, Peter Liu, a researcher at Google Brain, also realized the potential for AI to radically alleviate doctors' biggest problem with their EHR [7]. Leveraging the public MIMIC-III dataset (a collection of 39,597 publicly available, deidentified EHR records from the ICU of a tertiary hospital), Liu was one of the first to build a conditional language model from EHR-based note data that was not relegated to a specific modality (e.g. radiology imaging reports). Rather than simply using a generic generative language model, Liu conditioned his language model on the context from which the note originated due to the diversity of modalities that produce EHR note data. To do this, he annotated the model with the demographic data of the patient, current medications, laboratory results, and past medical history. This additional context was fed into the language model similarly to how special tokens in machine translation language models indicate the source language while learning. Liu then fed this context-enriched data into an enhanced Transformer-based model originally designed for machine translation, called Transformer with memory-compressed attention or T-DMCA, originally developed by Vaswani, et. al. [8]. He then sought in his experiments to develop a method to quantitatively measure the effectiveness of his language model through several metrics like perplexity per token, ROUGE scores, and metadata accuracy (e.g. when it generated an age for the patient, was the age correct based on the real underlying patient data). Liu's models performed badly when given only the current note as context to generate text, but began to perform very well when given additional context (such as demographic data, medications, lab results, and previous notes). Prior to being given these additional "hints", his models had a ROUGE-1 score of 19.8 and an age-guessing accuracy of 4.6%. When his model was given the additional hints, the ROUGE-1 score jumped to 40.5 and the age-guessing accuracy predictably increased to 97.4% accurate. Liu then went on to try to develop possible applications of his new model, and tested using it as a means of predictive text autocompletion, auto-correction of errors in medical text (fixing typos that the doctor wrote), and automatic meta-data population. All of which he found promising avenues of future research. In our paper here, we depart primarily from Liu's objective by building a language model solely from the note data itself with no additional context or structured data to help the model learn.

Outside of the clinical domain, we take further inspiration from other systems that solve similar problems in other domains, such as email writing. In 2016, Google announced a novel new feature for their two email products, Gmail and Inbox by Gmail, called "Smart Reply" [5]. Smart Reply was a deep learning effort to predict common responses to emails based on the context of the email chain. With Smart Reply enabled, the user is presented with a list of possible responses to their

emails a button click away. For common use-cases, this feature can radically decrease the amount of time a user needs to spend writing emails by "autocompleting" their messages for them. When launched to production, Google found that Smart Replies were used an astounding 10% of the time on mobile devices—an incredible amount of times considering the sheer number of users of Google's email offerings. To do this, they trained a neural deep learning architecture based on an LSTM-based seq2seq machine translation task at a massive scale to provide reliable smart reply suggestions that are "*always* high quality in language and content" and able to produce predictions efficiently within very strict latency requirements [4]. In order to train the model to solve this, they first preprocessed their corpus of English emails that represented their dataset. The resulting corpus had 238 million messages. After training, they then also developed an inference step to actually perform inference on unseen and new examples. To do this, after the completion of the training of the model, they feed a new original message $\mathbf{o}$ into the model and arrive at an output of the softmax'd probability distribution over the vocabulary at each timestamp. They then build the final resolution sequence by finding the most likely response token at each timestamp and inputting it back into the original model. They then ultimately implemented this system end-to-end in production to achieve their fantastic results. The "Smart Compose" system in 2018 built off of this foundation but took it further by presenting not just optional responses to the user, but actual autocompletion suggestions for them as they were writing the composition of the email [4]. To do this, Google learned off of not just the previous email's body, but the entire sequence of emails in the conversation in order to reliably be helpful to the end user. They then deployed this as well in a live system, their widely popular web-based email client, Gmail.

Finally, although in a different domain, we find the recent work done by Radford, et. al. at OpenAI in February, 2019, to develop advanced language models off of unsupervised data to be immensely helpful for the methods we will outline below [9]. Their full work, which they deemed too dangerous to release publicly, developed a multi-task unsupervised learning system based on a 1.5B parameter Transformer that they call GPT-2 [10]. They successfully confirmed their hypothesis that a generically trained unsupervised language model could be applied to many different tasks, such as question-and-answering without being trained off of the task-specific data. They tested this by analyzing their language model in a zero-shot setting on a multitude of tasks. Using this model, they achieved state-of-the-art results on 7 out of 8 NLP tasks they tested, including the Winograd Schema Challenge and LAMBADA (word prediction).

3 Approach

3.1 Data Pipeline

We pull the JSON representation of 48,957 veterinary medical record encounters that corresponded to visits made for either wellness, sick (urgent care), or surgical exams between January 1, 2019 and March 10, 2019.[1] The medical records contained a plethora of metadata about the patients and thee visit, but we scrubbed all of this information away. Since we were dealing with animal records only and we will not release the trained model publicly, we do not have to worry about the biggest stymieing force to AI research in healthcare: HIPAA and privacy laws. Each encounter is broken down into six sections: Intake, Subjective, Objective, Assessment, Plan, and Discharge. We provide a truncated example of an example record here for comparison in Figure 1.

During or after an exam, the veterinarian is recording all of these observations into these records. The first step in our pipeline was to preprocess all of this data so that it is usable within our model's pipeline for training. To do this, we manipulated the JSON representation of the records. Rather than separating the different sections of the exam into the Subjective, Objective, Assessment, and Plan sections, we merely concatenate all of them into one homogenous sequence of bullet points that we now call one of our encounters. Since each encounter was written in the form of bullet points to begin with, we replace the bullet points with new lines so that each new "point" remains on a single line. We then removed any blank sentences or sentences that simply reported a normal or not examined result like "normal", "not examined"/"ne", or "none." We then concatenated all of these encounters together into one singular long text file separating each encounter with a special `<END_OF_ENCOUNTER>` token that we inserted at the end of each encounter to separate them and so that our subsequent model can learn when to end encounters rather than blur into other ones. The

[1] The data comes from the Vetspire electronic health records platform for veterinarians.

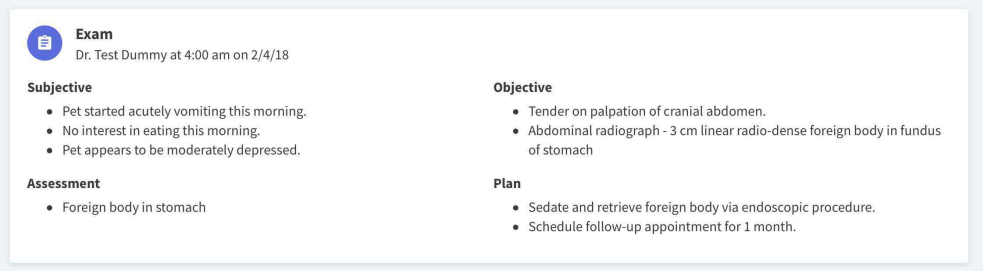

Figure 1: Sample encounter with SOAP sections.

resulting data-set yielded 570,760 individual bullet points across all encounters. We then split this dataset up into a training, dev, and testing set for our task below. The training set composed the first 456,608 bullet points. The dev and test set were each composed of 57,076 points. Unlike Liu, we did not perform any additional context-insertion or hinting within the dataset as the goal of our approach is to determine whether a model trained exclusively on the raw data absent any annotations can sufficiently model the domain to be useful in practice.

3.2 NMT Model

In this paper, we present two distinct approaches to solving our task—both rely on the latest advancements in NLP and deep learning. The first is to take inspiration from the Google "Smart Reply" and "Smart Compose" teams and to model this task as one of neural machine translation. We can describe this model as a seq2seq machine translation task. The source sequence is the sequence of embeddings derived above that corresponds to a bullet point in our dataset. The target sequence is the sequence of embeddings that correspond to the next bullet point. We first tokenize each bullet point into a list of words for processing by our word-level embeddings. After padding all of the bullet points into a sequence of tensors of length 256 (additional tokens after this mark were removed) composed of embeddings of size 64, we can then treat this as a translation task where our goal is to learn how we can translate the source sequence into the target sequence.

To do this training, we feed the network into a bidirectional LSTM encoder and a unidirectional LSTM decoder. We leveraging the initial starter code provided for us in assignment 4 as the initial basis for our implementation. We include a dropout layer as well. We preprocess our data into the source sequences and the target sequences and use these as our inputs to the model (built in pytorch).

3.3 Fine-tuned Language Modeling

Utilizing a different approach, we also explore solving this problem by fine-tuning a pre-trained language model with our unique dataset. Just recently, in February 2019, Open AI released a model pre-trained with 117M parameters (as compared to their not-released version of 1.5B parameters) [9]. Using this as our basis, we fine-tune this model on our own corpus of veterinary medical record data to produce a model bespoke to our use-case and domain (inspired by gwern, huggingface, and Neil Shepperd's work on fine-tuning language models for tasks such as poetry generation, whose code and infrastructure we base our implementation off of [14, 15, 19]).

The core of language modeling is when given a sequence of input tokens $(x_1, \ldots, x_n)$, we learn to predict the next token in the sequence, x_{n+1}. This essentially is the task of finding the token which maximizes the probability of appearing given the previously seen tokens.

In order to train the original base model, Radford, et al. leveraged a Transformer-based model that learned to model text data scraped from the web [16]. The pre-trained model was trained using 12 layers and a context size for the Transformer with a token size of 1,024. The Transformer architecture is a new method in which it models the contextual relationship between the words within a context by applying self-attention and computing the amount of attention that should be paid to each word in the context in determining the meaning of the word at hand [17]. This approach allows the model to be able to better understand the intercontextual relationships of words within text corpora and better be able to generate more coherent text over a long span of generation. Radford, et al.'s pretrained model

trained off of a rich and diverse collection of data compiled via web scraping. They scraped every web page on Reddit that was linked to that received more than 3 karma, compiling a dataset they refer to as WebText that contained 40GB worth of text [9]. Leveraging a large amount of computing power, they traind a 117 million parameter model off of this corpus. They provide this pretrained model for download, which we then use as the basis for our fine-tuning.

As a first step, we need to likewise encode and tokenize our own medical records data prior to training (the encoding step). Following the lead taken by Radford, et. al., we use Byte Pair Encoding (BPE) as a happy medium between word-level and character-level encoding [9]. BPE encodes words as a sequence of subword units in an attempt to facilitate out-of-vocabulary translation [11]. BPE is a compression algorithm first developed in 1994, but only recently used in AI [12]. Essentially, byte-pair encoding encodes the most common character pairs that appear within the same word as a single token. We leverage the implementation of BPE completed by OpenAI to perform the BPE encoding step [13]. The BPE approach works extremely well with our data set of medical records because a lot of the notes written by veterinarians will be a hodgepodge of abbreviations and terms, such as "no c/v/s/d" standing for "no coughing, vomiting, sneezing, or diarrhea." BPE is able to capture "c/" as a separate token that is unique from the "c" that appears normally within "coughing"—a huge difference that without would render our training almost moot. Likewise, if we went with a purely word-based approach, we would inevitably result in a plethora of UNK tokens, while a character-based encoding may be too unwieldy to train and not capture the link between different terms when they are concatenated together like the above example. Thus, the BPE approach works wonderfully well and is able to scale well to the varied language at play. After encoding, we can then move on to fine-tuning the pre-trained model with our dataset.

Fine-tuning a large and generic language model has been shown to be very successful in past attempts. Other researchers have been able to achieve state-of-the-art results by pre-training a very large language model, and then fine-tuning it by training on a specific supervised task afterwards. For example, a fine-tuned version of Google AI's BERT system accomplishes this successfully and currently ranks as the top system for various tasks including Stanford's Question and Answering (SQuAD) test [18]. BERT extends Open AI's first GPT-1 work by developing a bidirectional learning system. Since BERT had achieved so much success on various tasks, we first attempted to fine-tune BERT to our veterinarian medical record data by building an additional fine-tuning layer leveraging huggingface's open-source BERT reimplementation in PyTorch [19]. Unfortunately, we found that our attempts to fine-tune BERT lapsed qualitatively in our attempts. After training, it would still produce nonsensical utterances in its text generation. For example, when being prompted immediately with "P presents for recheck exam- P had a URI we treated with Clavamox" the system would generate the next sentence as "GI upset <1 second so sent with 0 to pursue HW"—a sentence that, despite being composed of tokens all found in the dataset and making roughly grammatical sense, lacks any medical relevance. Instead, we opted to fine-tune Open AI's GPT-2 model, which was just recently released, and provides a far more robust initial model—albeit superficially simpler in architecture and the public version only has 117 million parameters. The success found with GPT-2 may be due to a variety of causes, including the BPE encoder or even the biases in the original data trained.

In order to fine-tune GPT-2, we implement an additional final layer to learn off of our data-set, inspired by the approach of previous work done to fine-tune language models and GPT to specific tasks like poetry generation [14, 15, 19]. To do this, we feed the final hidden state of the Transformer pre-trained and feed that into an additional softmax layer with a cross-entropy loss. We use the Adam Optimizer. We have a batch size of 2, and otherwise use the same hyper-parameters as Open AI's GPT-2 original model training. We then first encode our medical records data as described above using BPE and then feed it into this model. This effectively fine-tunes the pre-trained GPT-2 to our very domain specific veterinary language.

We considered implementing our own training of a language model off of publicly available veterinary textbooks to capture the specific language of medical language over general-purpose language, but we found that this fine-tuning performed just fine, and lacking time and compute resources necessary to retrain a large language model from scratch, opted to simply fine-tune on top of a large existing model.

4 Experiments

4.1 Training and Modeling

Predictably, the original "baseline" implementation of this task as a neural machine translation problem fared poorly at capturing the essence of predicting veterinary medical records' sentences. Based only on a very small preceding window to "translate from," the NMT method could not adequately model the language. Nevertheless, to train the model, we fed the initial training data in the network, trained for 8 hours, and resulted in a final meager BLEU score of 1.19 when tested on the test data-set. While this is poor, it will serve as a good baseline. One of the most likely reasons for the low BLEU score is that we were only using the immediate previous sentence in our seq2seq model instead of the whole encounter. Like Google found, it will be a very hard task to learn prediction off of sequences of previous sentences when the length of each sequence is only 1 [4]. There is a large correspondence between the Subjective section and the Objective section that would not be captured by our baseline here that only looks at the immediate precursor. It is very hard to predict the "Oral" abnormal note when all you have to go off is the "Abdominal" note, but you could probably predict it after reading the Subjective section that included the word "halitosis."

Fine-tuning the language model fared much better. We trained our fine-tuning for 24 hours on a Microsoft Azure NV24 that had 4 M60 GPUs. It was able to reach a cross-entropy loss of 0.14 on our dataset of 48,957 encounters, and it was still decreasing slowly when we ceased training. Since GPT-2 was trained and modeled using TensorFlow rather than PyTorch, we likewise trained and modeled our implementation in TensorFlow as well.

4.2 Production Experiment

Given the success of fine-tuning the language model, we then went ahead and operationalized the results in a live environment with real veterinarians to test their usage in the field. We built a pipeline that took the final fine-tunely trained model and outputted the top result to the user for autocompletion. To do this, we built an API extension of the model that would take in a context of bullet points, tokenize it using BPE, and then feed that sequence of tokens into the model to generate conditional samples of text based on that line. We take the first line in the generated sample as the suggestion, decode it, and then have the API return the suggestion for the user. Prior to displaying this to the user, we also have to post-process the suggestion for quality control and personalization. If the suggestion was the `<END_OF_ENCOUNTER>` token, then no suggestion is shown. Further, we use the open-source language library for python, spacy, to perform named entity recognition on the sentence to find any names of dogs or cats that inadvertently were placed in the suggestion and replace it with the name of the patient currently being worked on. We also find, via find and replace, any and all pronouns and replace them with their appropriate version based on the patient the veterinarian is currently working on. Proper nouns that correspond to specific locations were removed altogether.

We then tie in this live pipeline into the user interface of the veterinary medical records application the data first originated from, which we base off of Google's "Smart Compose" system [4]. While veterinarians are writing various portions of their notes, if there is a pause in typing of more than 250ms detected at any point, then it calls for a suggestion from the API based on the all of the previous sentences already written in that encounter and presents it to the user as a grayed out suggestion as shown in Figure 2. All the user needs to do then is to hit the `TAB` character on their keyboard to have the text autocomplete with the suggestion. In order to ensure that our model can sample at the appropriate pace of users typing speeds, we keep the model "hot", by keeping the preloaded model and session continuously running, awaiting new input.

We recall that Google's "Smart Reply" system was used in 10% of times on mobile when released publicly [5]. Our autocomplete suggestions, after 48 hours of being used in production, was used at least once in 12.46% of encounters! We understand that the domain of email replies may be larger than the domain of medical language (although disputable), we take this as an astounding success of our model, and the best validation we can have that the model was able to properly model veterinary medical records language.

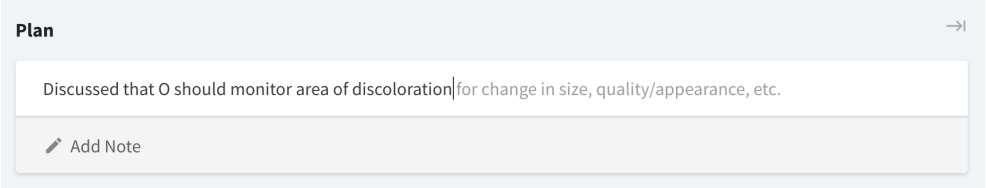

Figure 2: Example of the autocompletion suggestion in the interface that veterinarians use to compose their medical records. The veterinarian would hit TAB to complete this suggestion.

5 Analysis

Based on the production testing described above, we were able to achieve fantastic success in our model with veterinarians actually finding it useful in production use cases. This production success is the best validation possible that the fine-tuned language model was able to accurately and helpfully understand the domain and be able to discover what best to suggest to live users as they write up their own notes. We can investigate some of the more interesting pieces of generated text based on prompts to gain further insight into the behavior of our model below.

During a dental procedure, we discovered our model was able to helpfully suggest a full paragraph worth of dental discharge instructions for the veterinarian that reads with nearly perfect English and great contextual accuracy to the domain of dental discharge notes. After being prompted with "Post-Dental Instructions:", the model suggested the following text:

```
Your pet may become sleepy for a couple days after the anesthesia.
I recommend soft food after the dental because your pet's mouth may
be sensitive.  After feeling better, the regular diet may be given.
```

We discovered that the model was being often used by veterinarians while they were writing discharge instructions, as the model must have been able to better grasp the very similar discharges that veterinarians would write after usual dental or other procedures. As another prime example, the model could predict the following text after being prompted with "Pro-heart Vaccine reactions:"

```
Your pet received vaccines and/or pro-heart injection (0.3 ml) which
is heartworm prevention that persists for 6 months.  Mild side
effects of vaccines are possible, so please watch for any signs
of a severe allergic reaction that can occur within several hours
of vaccination, which may include vomiting, diarrhea, itchy skin and
hives, difficulty breathing, facial swelling, or collapse.
```

The model was not just able to model discharge instructions very well, but it was also able to successfully autocomplete during the regular SOAP portion of the exam as well. When prompted with "P presented for limping on left limb,", the model replied beautifully with, "`P does not seem to care for the leg.  P is still walking on it, wont seem to be putting weight on right limb.  O states P has had some favoring back in the room however.`" In this specific example, we discover that our language model could not just model discharge instructions successfully, where the language is most similar to other language one might find on the internet, but also veterinary specific lingo and vernacular like using 'P' for patient and 'O' for owner, which the model discovered all on its own. In fact, this example almost reads like it was a veterinarian note itself. Other examples of subjective suggestions include, "P in great condition at today's visit. Owners have no big complaints at this time. Previous diagnosis of HWD. To use medicated shampoo. If no improvement will consider dental, mass removal. To keep us closely updated with any changes at home."

Moreover, we discovered that the model could even predict medically-relevant diagnoses in some cases. For example, when prompted with the final line being, "Rec'd adding in a dental to control pruritus", the model suggested, "1) Grade 3 dental diseases 2) Otitis externa d." The model did, however, struggle to provide reliable suggestions during the "Objective" section. Incorrect suggestions included, "Cherry eye - cherry eye surgery" or "UTD Rabies and DHPP, ok to go to doing cherry eye surgery", neither of which make sense medically or contextually. We would suggest that the more

patient and result-specific area of the "Objective" section posed a problem for our language model since it was harder to predict the specific and short notes that a veterinarian would write that was often numeric in nature.

Despite these limitations, the model showed adeptness throughout the varied medical records. In addition to discovering diagnoses, recording histories, and writing discharges, it was also able to record treatments such as vaccinations when entering the treatment section of the record, after being prompted with a normal wellness exam with various words such as "o came in for vaccines" placed around the record, the model was able to give suggestions such as: "Rabies SQ RR - 1yr DHPPL SQ L - 1yr Bord PO - 1yr", which models the exact type of notes that a veterinarian would write to record their administration of these three vaccines and the specific areas of the body (in abbreviations) where they did so. It was also able to accurately model animal medicine specific terms, such as suggesting the following when being prompted with "EENT:", ''`Otitis externa secondary to primary conditions (e.g., over cleaning vs exploration) due to excessive debris throughout ear canals and very mild debris`''.

6 Conclusion

We conclude that our model was able to accurately and successfully model the domain of medical records and successfully be used in a production system for veterinarians. Veterinarians selected our autocompletion suggestions, trained after fine-tuning Open AI's GPT-2 language model, 12.46% of the time. It was able to learn the most common areas of writing very well that had the most repetition in the data-set (such as discharge notes, recommendations, or histories). Yet, it also adapted to various other areas of the medical record with remarkable success.

The primary limitation of our current model is that while it was shown to adapt well to the domain at hand, it failed to properly give suggestions during the "Objective" section, which is composed more of patient-specific details. Given more data, compute resources, and training time, we believe these errors could be solved. Additionally, there may be additional avenues to explore with regards to the problems our model could solve beyond autocompletion, like typo fixing, or, even, treatment suggestions. All in all, our model shows that fine-tuning large-scale general language models to very domain-specific language can be amazingly successful, and we encourage more research in this field.

Additional Information

Mentor: Anand (anandd@stanford.edu)

References

[1] Sinsky, Christine; Lacey Colligan, MD; Ling Li, PhD; Mirela Prgomet, PhD; Sam Reynolds, MBA; Lindsey Goeders, MBA; Johanna Westbrook, PhD; Michael Tutty, PhD; George Blike, MD. "Allocation of Physician Time in Ambulatory Practice: A Time and Motion Study in 4 Specialties." *Annals of Internal Medicine* 165 (11), pp. 753-776.

[2] Arndt, Brian G., John W. Beasley, MD, Michelle D. Watkinson, MPH, Jonathan L. Temte, MD, PhD, Wen-Jan Tuan, MS, MPH, Christine A. Sinsky, MD and Valerie J. Gilchrist, MD. "Tethered to the EHR: Primary Care Physician Workload Assessment Using EHR Event Log Data and Time-Motion Observations." *Annals of Family Medicine* 15:5, pp. 419-426.

[3] Friedberg MW, Chen PG, Van Busum KR, et al. "Research report: factors affecting physician professional satisfaction and their implications for patient care, health systems, and health policy." Santa Monica, CA: Rand Corporation, 2013.

[4] Wu, Yonghui. "Smart Compose: Using Neural Networks to Help Write Emails." *Google AI Blog*, May 16, 2018.

[5] Kannan, Anjuli and Karol Kurach and Sujith Ravi and Tobias Kaufman and Balint Miklos and Greg Corrado and Andrew Tomkins and Laszlo Lukacs and Marina Ganea and Peter Young and Vivek Ramavajjala. "Smart Reply: Automated Response Suggestion for Email." Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) (2016)

[6] Marmor, R.A., B. Clay, M. Millen, T.J. Savides, and C.A. Longhurst. "The Impact of Physician EHR Usage on Patient Satisfaction." *Applied Clinical Informatics* Vol. 9, no. 1, pp. 11-14 (2018).

[7] Liu, Peter J. "Learning to Write Notes in Electronic Health Records." CoRR (2018).

[8] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Lukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. "Tensor2tensor for neural machine translation." CORR (2018).

[9] Radford, Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever. "Language Models are Unsupervised Multitask Learners" (2019)

[10] Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units." *Proceedings of the 54th Association of Computational Linguistics*, pp. 1715-1725 (2016).

[12] Gage, Phillip. "A New Algorithm for Data Compression." *C Users Journal*, Vol. 12, no. 2, pp. 23–38, (1994).

[13] "Code for the paper 'Language Models are Unsupervised Multitask Learners'." https://github.com/openai/gpt-2.

[14] Gwern, "Finetuning the GPT-2 Small Transformer for English Poetry." https://www.gwern.net/RNN-metadata#finetuning-the-gpt-2-small-transformer-for-english-poetry-generation

[15] Shepperd, Neil. "Fine-tuning GPT-2" https://github.com/nshepperd/gpt-2

[16] Radford, A., K. Narasimhan, T. Salimans, and I. Sutskever. "Improving language understanding by generative pre-training" (2018).

[17] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. "Attention is all you need." In *Advances in Neural Information Processing Systems*, pp. 5998–6008 (2017).

[18] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805 (2018).