Automatically Generated Summaries of Video Lectures May Enhance Students' Learning Experience

Hannah Gonzalez*, Jiening Li*, Helen Jin*,
Jiaxuan Ren*, Hongyu Zhang*, Ayotomiwa Akinyele*,
Adrian Wang, Eleni Miltsakaki, Ryan S. Baker, Chris Callison-Burch
University of Pennsylvania
hannahgl,jiening,helenjin,rjx,hz53,tomiwa,kelmp,elenimi,rybaker,ccb@seas.upenn.edu

Abstract

We introduce a novel technique for automatically summarizing lecture videos using large language models such as GPT-3 and we present a user study investigating the effects on the studying experience when automatic summaries are added to lecture videos. We test students under different conditions and find that the students who are shown a summary next to a lecture video perform better on quizzes designed to test the course materials than the students who have access only to the video or the summary. Our findings suggest that adding automatic summaries to lecture videos enhances the learning experience. Qualitatively, students preferred summaries when studying under time constraints.

1 Introduction

Video lectures have been an important part of scaled online courses and flipped classrooms for several years, and have become widely used for an increasingly larger range of courses as a substitute for students unable to attend class due to the COVID-19 pandemic (van Alten et al., 2020). Past research in human-computer interaction aimed to improve educational videos via interactive transcripts, word clouds, keyword search, and highlight storyboards (Kim et al., 2014), or by segmenting the videos to present highlight moments with snapshots and transcripts (Yang et al., 2022). Others have created video digests that are organized into a textbook-like format with chapters, titles, and sections with text summaries (Pavel et al., 2014). Pavel et al. (2014)'s system provides an authoring interface that allows video authors to manually write textual summaries of a video themselves or to send the video to a crowdsourcing service to have summaries written. Textual summaries are believed to be effective in helping students review course materials. For example, Shimada et al. (2017) find that students using summaries of slides for preview have higher pre-quiz scores and spend less time, compared to students previewing original learning materials.

In this work, we investigate the feasibility of automatically summarizing lecture videos' transcripts using recent advances in large language models such as GPT-3 (Brown et al., 2020). We are encouraged by recent research in natural language processing demonstrating that people often prefer GPT-3 generated summaries over other methods of automatically generated summaries for news (Goyal et al., 2022).

The availability of high-quality automatic summaries would allow their use in a wide range of online courses. In this paper, we first detail our method for creating an automatic summarizer of video lectures. Then, we report a controlled user study that we conducted with around 100 university students to investigate whether the automatic summaries were an effective study aid.

2 Automatic Video Lecture Summarizer

We designed an automatic lecture summarizer that leverages recent advances in large language models (LLMs). Large language models are trained on massive language datasets, thus creating rich representations of language content. These models have been especially successful in generating fluent responses in a variety of natural language generation tasks with zero or few-shot prompting (Brown et al., 2020; Liu et al., 2023), instruction following (Sanh et al., 2021; Mishra et al., 2022; Bach et al., 2022), or fine-tuning (Howard and Ruder, 2018; Wei et al., 2021).

It is well-known, albeit not well-understood, that the quality of GPT-3 language generation is affected by the given prompt even if the differences in the way the prompt is articulated are not semantically very different. It is, also, unclear for which tasks it is sufficient to give prompts to achieve suc-

* These authors contributed equally.

Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023), pages 382–393
July 13, 2023 ©2023 Association for Computational Linguistics

cessful GPT-3 responses or whether there is a need for fine-tuning a model. For these reasons, we experimented with three approaches to generating lecture summaries.

In all of our experiments, the input provided to the model is retrieved by transcribing video lectures. For the video transcription, we use Amazon Transcribe. The transcribed video content is then split into chunks of a maximum of 4,000 tokens. This is the maximum length that can be encoded by GPT-3. The final summary of the lecture is the aggregate of the summaries of its parts. For each chunk, we retain the corresponding start time in the video.

(1) Zero prompt summarization. In the first approach, for each retrieved chunk we generated a summary providing the following prompt:

> I am a summarization bot. If you give me text, I will provide a textbook-like summary.
>
> Text: <transcript_chunk>
> Summary:

In this condition, the summary is generated without any training. We call this approach the "zero prompt" condition. Below we provide an input example of a raw transcript from a Natural Language Processing lecture given in an Artificial Intelligence university course.

Sample input to summarizer:

> So uh if we're just talking about, we were just talking about input sequences for our problem with dimensionality, then let's imagine that we've got an input sequence of like 10 And we've got of okay, let's imagine we have a vocabulary size of 100,000. then we've got 100,000 to the power of 10 different possible sequences, word combinations. So that ended of itself is going to be 10 to the power of 50 parameters like unimaginably large, way more then? The number of ida adam is in the universe? So too high to estimate from data.

Summarizer output:

> If we have a vocabulary size of 100,000 and an input sequence of 10, there are 100,000 to the power of 10 different possible sequences. This is too high to estimate from data.

As can be seen in the summarizer output above, the zero-shot summarizer removes filled pauses like *um* and *uh*, eliminates false starts like the repetition of *let's imagine*, deletes misrecognized phrases like *the number of ida adam* (which should be *the number of atoms*), and concisely summarizes the idea in the transcript chunk.

In Appendix A we provide details about the model variant and parameters that we used.

(2) Few shot prompt summarization. In the second approach, we enriched the prompt by providing ten prompt-summary pairs as context to the current prompt.

In this approach, we provide a few training examples to the GPT-3 model and we call it the "few shot prompt" condition. The structure of this few-shot prompting is as follows:

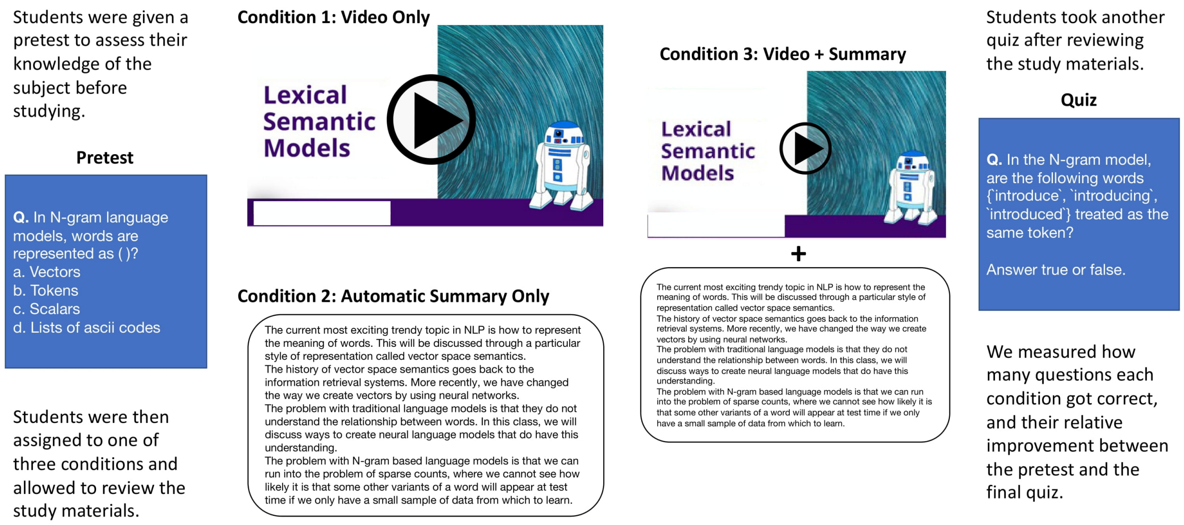
> I am a summarization bot. If you give me text, I will provide a textbook-like summary without repeating past summaries or describing the speaker.
>
> Text: <chunk from 10 prompts ago>
> Summary: <summary of that chunk>
> ...
> Text: <chunk from 9 prompts ago>
> Summary: <summary of that chunk>
> ...
> Text: <chunk from 1 prompt ago>
> Summary: <summary of that chunk>
>
> Text: <current_transcript_chunk>
> Summary:

In this condition, we modified the prompt by adding to it "without repeating past summaries or describing the speaker." This modification prevented the summarizer from a) repeating past summaries and b) starting summaries with statements like *The speaker is discussing [topic].* To further reduce repetition errors, we included in the summarizer a step to check if the current summary output matches any of the previous summaries. If so, GPT-3 would be prompted to generate a new output on the same prompt.

We observed that there were several advantages to including previous chunks and their summaries as part of the input. First, they provide useful context for subsequent summaries to remain topical. Second, transcription errors are not always uniform across chunks. For example, the term *n-gram* is misrecognized in the following chunk of the transcript.

> What we what we're doing is basically just constructing a table. Look I wanted to say here's a sequence. What's the probability of the next word? Just looking up at the table. And the trick for these **engram** based language models was how do we deal with unseen sequences? So how do we deal with new combinations of n words that were never that never occurred in our training So we did things like smoothing, we did things like interpretation. We did back off too small, the two smaller and smaller sequences.

Due to the context given previously, the summarizer provides a correction in its output, as can be



Students were given a pretest to assess their knowledge of the subject before studying.

Pretest

Q. In N-gram language models, words are represented as ( )?
a. Vectors
b. Tokens
c. Scalars
d. Lists of ascii codes

Students were then assigned to one of three conditions and allowed to review the study materials.

Condition 1: Video Only

Condition 2: Automatic Summary Only

The current most exciting trendy topic in NLP is how to represent the meaning of words. This will be discussed through a particular style of representation called vector space semantics.
The history of vector space semantics goes back to the information retrieval systems. More recently, we have changed the way we create vectors by using neural networks.
The problem with traditional language models is that they do not understand the relationship between words. In this class, we will discuss ways to create neural language models that do have this understanding.
The problem with N-gram based language models is that we can run into the problem of sparse counts, where we cannot see how likely it is that some other variants of a word will appear at test time if we only have a small sample of data from which to learn.

Condition 3: Video + Summary

The current most exciting trendy topic in NLP is how to represent the meaning of words. This will be discussed through a particular style of representation called vector space semantics.
The history of vector space semantics goes back to the information retrieval systems. More recently, we have changed the way we create vectors by using neural networks.
The problem with traditional language models is that they do not understand the relationship between words. In this class, we will discuss ways to create neural language models that do have this understanding.
The problem with N-gram based language models is that we can run into the problem of sparse counts, where we cannot see how likely it is that some other variants of a word will appear at test time if we only have a small sample of data from which to learn.

Students took another quiz after reviewing the study materials.

Quiz

Q. In the N-gram model, are the following words {`introduce`, `introducing`, `introduced`} treated as the same token?

Answer true or false.

We measured how many questions each condition got correct, and their relative improvement between the pretest and the final quiz.

Figure 1: We introduce a new method for creating automatic summaries of lecture video transcripts, and perform a user study with 106 students to determine whether supplementing videos with the summaries enhances their learning.

seen in the output shown below (but not for the word *interpolation* that remains incorrectly transcribed as *interpretation*):

> The trick for **n-gram** based language models is how to deal with unseen sequences. This is done with smoothing, interpretation, and back off to smaller sequences.

Table 2 in the Appendix gives more examples of the automatic summaries produced by our few-shot model.

(3) Fine-tuning GPT-3. In the third approach, we experimented with fine-tuning GPT-3 to perform lecture summarization. We manually edited the output of our few-shot model described above in order to provide annotated examples for fine-tuning. By this process, we obtained 114 prompt/output pairs which we then used to fine-tune a summarization model. When fine-tuning a GPT-3 model, we no longer need to provide prompts like we did in the previous two approaches.

The motivation behind experimenting with different approaches to summarizing transcribed video lectures was to identify a model that is likely to yield quality summaries. Through a series of informal evaluations of the three types of outputs, we observed that the fine-tuned model produced summaries that were more consistent in style and contained less repetition than the zero-shot and few-shot models. Table 3 in the Appendix gives examples of the automatic summaries produced by our fine-tuned model. As our main interest in this study is to evaluate whether adding summaries to video lectures yields learning benefits to students' review of course materials, we did not perform a formal evaluation of the three approaches to automatic summarization. Instead, we opted to conduct a controlled study to evaluate the learning benefits of summarization in three course reviewing conditions. We report this evaluation study in the next section.

3 Evaluation Study

In this section, we report a controlled study that we conducted with the goal of evaluating the potential benefits of offering students an automatic summary of transcribed video lectures. In what follows, we describe the participants of the study, the testing conditions, and the results.

Participants. We recruited 106 undergraduate and Master's students who were taking an Artificial Intelligence course in Fall 2022. Students were given extra course credit for their participation.

Study design. We evaluated student performance on materials that students reviewed for two upcoming topics in the course presented in video lectures. These consisted of two short, prerecorded lecture videos on Lexical Semantic Models (10 minutes) and Stochastic Gradient Descent (12 minutes). For each topic, we evaluated the three learning conditions listed below.

Testing conditions

1) Reviewing only the lecture video, 2) Reviewing only the automatically generated summary, 3) Reviewing both the video and the automatic summary.

Cond.	N	Pre-test	σ	Post-test	σ	Δ
Sum.	56	62%	1.03	73%	1.4	17.7%
Video	39	67%	0.9	79.7%	1.1	18.5%
V+S	48	66%	0.9	82%	1.2	24.4%

Table 1: Mean correctness on the pre-test quizzes, and mean correctness on the quiz after reviewing the study materials for students in each of our three learning conditions for both lectures: Summary, Video, and Both.

All students were randomly assigned to a different learning condition for each topic. Many participants reviewed both lecture topics. For the second round, they were assigned to a different learning condition.

Prior to reviewing the course materials, students were given pre-test quizzes with four questions for each topic to test their initial understanding of the concepts. The answers were not shown to the students. After the pre-test, students reviewed the course materials using the materials associated with their randomly assigned learning condition. Finally, they answered a 10-question quiz on the material that they had reviewed. These included the 4 questions from the pre-test, plus 6 previously unseen questions. The quiz questions consisted of a mix of True/False questions and multiple-choice questions. The quiz questions are given in Appendix D.

The students were not given a time constraint for reviewing the materials. However, once they started the quiz, they were no longer allowed to review the materials. Table 1 summarizes the students' performance under the different learning conditions. We calculated the relative percentage point increase as follows:

$$\Delta = \frac{\text{Post score (\%)} - \text{Pre score (\%)}}{\text{Pre score (\%)}}$$

The mean correctness on the pre-test quizzes is below 70%. After reviewing the learning materials, the condition in which students demonstrate the smallest improvement is condition (2) with only access to the automatic summaries improve: a relative improvement of 17.7% or an 11% absolute improvement. Students who reviewed only the videos (condition 1) have a relative improvement of 18.5% (12.7% absolute). Students who reviewed both the videos and the automatic summaries have a relative improvement of 24.4% (16% absolute). A finer-grained breakdown of the students' performance on the quizzes for each lecture video is given in Appendix F.

We conducted a paired t-test to determine if there was a significant difference in the test correctness scores before and after the video+summary intervention. The results showed a calculated t-statistic of 2.12 and a p-value of 0.045 for the Lexical Semantic Lecture, as well as a calculated t-statistic of -4.16 and a p-value of 0.0003 for the Stochastic Gradient Descent Lecture. These findings indicate a significant difference between the means. Although we cannot conclusively determine that the video+summary approach is the most effective learning condition among those tested (as indicated by the Kruskal-Wallis test result of $H(2) = 2.13$ and a p-value of 0.34), we can observe that the results show a positive trend in the desired direction.

3.1 Qualitative student feedback

In order to examine the potential impact of time constraints on learning, we solicited feedback from an additional group of students that learned under a two-minute time constraint. They were allowed to learn from both the video lecture and the summary within the given time frame. After the experiment, we asked the students to fill out a qualitative feedback survey about their study methods, specifically if they utilized the summary, video, or both.

Overall, we found that under timed conditions, students tended to use summaries over video lectures when both were available just as they would do when studying before an exam when time constraints make summaries more useful. We report three representative quotes from the student responses:

"Summaries are helpful to get an overview of lecture."

"Used mainly the timestamps and the summary, didn't pay too much attention to the video itself."

"I initially did not look at the lecture summarizer because the material was new to me and as a result it seemed better to take in a larger quantity of material with new details. However, over time, as I began to get confused or did not recall all details about the lecture I started looking at the summaries. This is where I felt the lecture summaries were particularly valuable - to reinforce details about the lecture that I might have overlooked. Initially, without the context of having already watched the lecture, the summaries were not useful, but with this context existing as s sort of partial lattice structure in my head, the summaries became useful for filling the gaps that were missing from that structure."

4 Conclusion

Our work shows that students reviewing both the lecture video and the automatically generated summary have a performance improvement from pretest to post-quiz. This suggests that accompanying lecture videos with automatically generated summaries does improve the studying experience. As online learning becomes more ubiquitous, incorporating automatically generated summaries with videos can enhance students' overall learning experience.

5 Limitations

Our user study tests students on only two short lecture videos which are pre-recorded and carefully edited. Future work should test the efficacy of the summaries under a wider range of conditions: prerecorded videos versus live lectures, lectures and summaries of different lengths, and a wider range of topics and disciplines.

Overall, our experiments compare three different conditions. Adding other conditions might have shed light on the relative value of automatic summaries. For instance, if we limit the time available for participants to prepare before taking the quiz, and at the same time track the amount of time spent on summaries and/or videos, then that could give better insights into how students would utilize the two sources differently with limited time constraints. Finally, we could also contrast the usefulness of summaries versus transcripts.

Acknowledgements

This research is based upon work supported in part by the DARPA KAIROS Program (contract FA8750-19-2-1004), the DARPA LwLL Program (contract FA8750-19-2-0201), the IARPA HIATUS Program (contract 2022-22072200005), and the NSF (Award 1928631). Approved for Public Release, Distribution Unlimited. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, IARPA, NSF, or the U.S. Government.

We would also like to thank Bryan Li and Rotem Dror for their helpful suggestions.

Finally, we extend our sincere appreciation to Professor Andrew Head for his guidance in experiment design and insightful discussions.

References

Stephen Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-david, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Fries, Maged Alshaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-jian Jiang, and Alexander Rush. 2022. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104, Dublin, Ireland. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. News summarization and evaluation in the era of GPT-3. *arXiv preprint arXiv:2209.12356*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Juho Kim, Philip J Guo, Carrie J Cai, Shang-Wen Li, Krzysztof Z Gajos, and Robert C Miller. 2014. Data-driven interaction techniques for improving navigation of educational videos. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 563–572.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. 2014. Video digests: a browsable, skimmable format for informational lecture videos. In *UIST*, volume 10, pages 2642918–2647400. Citeseer.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Atsushi Shimada, Fumiya Okubo, Chengjiu Yin, and Hiroaki Ogata. 2017. Automatic summarization of lecture slides for enhanced student preview: Technical report and user study. *IEEE Transactions on Learning Technologies*, 11(2):165–178.

David C.D. van Alten, Chris Phielix, Jeroen Janssen, and Liesbeth Kester. 2020. Self-regulated learning support in flipped learning videos enhances learning outcomes. *Computers Education*, 158:104000.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Saelyne Yang, Jisu Yim, Juho Kim, and Hijung Valentina Shin. 2022. Catchlive: Real-time summarization of live streams with stream content and interaction data. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

A Model Details

We use the ‘text-davinci-002’ version of GPT-3 for our zero-shot and few-shot experiments, and as the basis for our fine-tuned davinci model. We use these parameters:

- Temperature: .6
- Frequency penalty: .7
- Presence penalty: .7
- Max tokens: maximum possible

To compute the maximum possible tokens for each API call we made to the model, we start with the total number of tokens that the model can process (4000 tokens for ‘text-davinci-002’, 2048 for our fine-tuned model) minus the number of tokens in the current prompt. We used OpenAI’s ‘GPT2TokenizerFast’ (from huggingface-transformers) to count tokens.

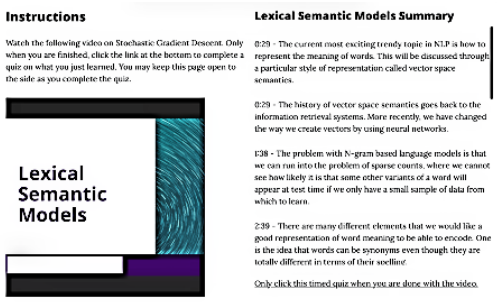
B Example Summaries

Table 2 gives example summaries from 13 consecutive transcript chunks from a lecture on Neural Network Language Models given in an Artificial Intelligence course. This output is produced by our few-shot model. In the few-shot model, there are many repetitive outputs with several of the summaries beginning at *The speaker is discussing*.

Table 3 gives example summaries from 15 consecutive transcript chunks from a lecture on Vector-Space Semantic Models given in an Artificial Intelligence course. This table shows outputs from our fine-tuned model. We can observe that the repetitions in the summary are gone and the style of the summaries has improved.

C User Interface

Figure 2 shows the interface template for the video-and-summary condition group in the user study, where we have instructions and a lecture video on the left side, and a summary with timestamps on the right side followed by a quiz link. Students can scroll down on the lecture summary.



Instructions

Watch the following video on Stochastic Gradient Descent. Only when you are finished, click the link at the bottom to complete a quiz on what you just learned. You may keep this page open to the side as you complete the quiz.

Lexical Semantic Models Summary

0:29 - The current most exciting trendy topic in NLP is how to represent the meaning of words. This will be discussed through a particular style of representation called vector space semantics.

0:29 - The history of vector space semantics goes back to the information retrieval systems. More recently, we have changed the way we create vectors by using neural networks.

1:38 - The problem with N-gram based language models is that we can run into the problem of sparse counts, where we cannot see how likely it is that some other variants of a word will appear at test time if we only have a small sample of data from which to learn.

2:39 - There are many different elements that we would like a good representation of word meaning to be able to encode. One is the idea that words can be synonyms even though they are totally different in terms of their spelling.

Only click this timed quiz when you are done with the video.

Figure 2: Video-and-Summary Interface

D Pre-Quiz and Post-Quiz questions

Lexical Semantic Pre-Quiz Questions

1. In N-gram language models, words are represent as
   (a) Vectors
   (b) Tokens
   (c) Scalars
   (d) Lists of ASCII codes

2. In the N-gram model, are the following words (introduce, introducing, introduced) treated as the same token? (T/F)

3. What is the LEAST important factor that we should consider while encoding word meanings?
   (a) Words have synonyms
   (b) Words that have opposite meanings
   (c) Words that reflect different semantics roles
   (d) The number of word occurrences

4. Semantic representation will not involve encompassing syntactic positions of the words. (T/F)

Raw Transcript Chunks	**Automatic Summaries**
I feel a little bit like I'm landing an airplane coming in for the end of the semester filled with hundreds of people. But other than that. All right. How you doing? Thank you. Gave me a linchpin for the here. I appreciate it. Thank you. Yes. Yeah. It's a to cope with the mask of the students. Good. Thank you. Thank you. You too. Right. Yeah. All right. So now we're on to the very last topic of the class. My personal favorite natural language processing.	The speaker is discussing their favorite topic, natural language processing. They mention that they feel like they are landing an airplane when coming in for the end of the semester. The speaker thanks the person they are talking to and mentions that it is difficult for students to cope with mask wearing.
So uh we call it natural language processing to distinguish it from programming languages, natural language. Um and there's lots and lots of different elements of this field. So I'm gonna walk you through one particular topic. So we'll talk about um how we've been, how we've adapted to use neural networks to do the problem of language modeling. And then we'll talk more generally about how we can represent the meaning of words. So how do we come up with a computer interpret herbal representation of word meaning?	The speaker is discussing natural language processing and how it is different from programming languages. The speaker talks about how neural networks can be used to do the problem of language modeling. The speaker also talks about how computer representations of word meaning can be created.
Embedding and stores this in a matrix ah where we have the words in the vocabulary and each Word embedding of length M. So that's that 10, 30, 60 kind of length. Um And in this old paper they initialized this with singular value decomposition. But detailed doesn't really matter that much anymore. And the neural architecture that they use, I was just feed forward neural net. Okay, but here's my cartoon of what their neural network looks like. We've got our input words here. So we've got some number of input words, we want to predict the next word.	The neural network they use is a feed forward neural net. The input words are fed into the network, and the goal is to predict the next word.
And then we have a bunch of hidden layers and then we aggregate all the weights together into a weight matrix for each hidden layer and we have an output layer that we usually push through a softmax function. That changes the uh numbers here that are real valued numbers into a probability distribution. And that output layer, the number of nodes in the output layer can correspond to the number of classes that we have in our prediction problem. Okay. And then the math underlying, it is not that much different than we had for the logistic progression classifier.	Feed forward neural nets work. The math underlying the neural net is similar to the logistic progression classifier.
So the main difference is instead of taking the dot product between a weight vector and the input vector, we now take a matrix multiplication between the input vector X. And the weight matrix for that entire hidden layer. Right? But we still have the bias term. We still pump it through an activation function which could either be the sigmoid just like in logistic regression or it could be one of another uh set of things like rectified linear units or tangent. All right. And so that that then gives us the output of pin layer.	The main difference between a feed forward neural net and logistic regression is that with a feed forward neural net, we take a matrix multiplication between the input vector and the weight matrix. We still have the bias term and we still pump it through an activation function.
And then each hidden layer could act as the input to the subsequent hidden layer or to the final air. So each of these transmits uh its own output to the next player, just like the input to this first layer. And so each layer has its own set of weights. So these are the parameters of the model. And finally we put things through the softmax operation. Right.	Each hidden layer in a feed forward neural net transmits its output to the next layer. The weights for each layer are the parameters of the model. Finally, the output is put through a softmax operation.
And so last time we talked about training the networks Arianna, we use a cross entropy loss function, just like with logistic regression, we compute the gradients but we use uh the computation graphs and the back propagation algorithm. Okay, so how can we use these neural networks to do the problem of language modeling? So you'll recall from your language modeling homework that one of the goals of language modeling is to estimate the probability of some sequence of words that form a sentence.	Neural networks can be used for language modeling. They talk about how the goal of language modeling is to estimate the probability of a sentence.
So if we've got a sequence of words uh where we have them index from word one up towards the end been we can use the chain rule for probability to say the probability of each next word, depends on all the previous words. And the simplification that we've made in our language modeling homework, in our n gram based language modeling homework was to make a Markov assumption where we said the probability of the next word only depends on a short sequence of words in its history.	The probability of each next word in a sentence depends on all the previous words. The simplification that was made in the language modeling homework was to make a Markov assumption where the probability of the next word only depends on a short sequence of words in its history.
So it doesn't depends on the whole world history, just a short sequence. And the advantage that that had, was it allowed us a straightforward mechanism for estimating the probabilities. Alright, so once we make that Markov assumption, estimating the probabilities of each next word, given that limited history is really just a matter of going through and counting up that sequence of those and words and or and divided by the total number of words. Words with that preceding history plus any other words. Right. And so that would just be the maximum likelihood estimate.	The Markov assumption allows for a straightforward mechanism of estimating probabilities. Estimating the probability of each next word is just a matter of counting up the sequence of words and dividing by the total number of words with that preceding history.

Table 2: Example summaries generated by our few-shot summarizer

Raw Transcript Chunks	**Automatic Summaries**
Any questions about the homework assignments, um, or the reading from, for the Ngrams? Nope. Okay. Um, so as friendly reminders, homeworks do at midnight, you have a total of five free late days for the course. Uh, you could use them at any time without asking the maximum number of late days you could use on any assignment is two. So, um, but I also recommend that maybe you don't. Uh, use them early if at all possible, because then you're just stuck in this thing with being a couple days behind on the next homework.	There are five free late days for the course, which can be used at any time. However, it is recommended that students do not use them early if at all possible.
So they're a blessing and a curse. Um, the next homework will be out shortly. I'm meeting up with one of the TAs to look over the revisions of it from last year, this afternoon. So with any luck, we'll get up, get it up this afternoon, even before, uh, homework number three is due. So this homework will be, uh, implementing the style of, uh, vector space models that we talked about, um, on Monday, which are the, uh, uh, long sparse vectors that you get via accounting. And then interesting transformations on the counts.	The next homework will be implementing vector space models using the counts from a corpus.
Like the TF IDF, the term frequency, inverse document frequency one, and also the positive pointwise mutual information, which we didn't cover in class, but which is included in the textbook in chapter six. So. Please do read that section. Um, and, uh, we'll sh we'll like, have you, uh, analyze a Corpus of Shakespeare's writings? So you'll be able to say like, uh, for this term by document matrix, pull out the column vectors, representing the documents and compare, uh, pairs of Shakespeare's plays.	The term frequency, inverse document frequency, and positive pointwise mutual information are all types of vector space models. The class will analyze a corpus of Shakespeare's writings using these techniques.
So, um, when people in the English department are studying Shakespeare, they categorize, uh, his plays into like dramas, comedies and histories. So it might be interesting to see whether, uh, the plays that are in those conceptual categories established through literature, um, have a higher co-sign similarity with each other than with the other categories. So that's one potential analysis that you could do, um, and then read the textbook. Chapter six, that'll be the quiz that'll, uh, be released this week and will be due again on Monday at midnight. Okay.	One potential analysis is to see whether plays in the same conceptual category have a higher co-sign similarity with each other than with the other categories.
So, uh, uh, last time we were talking about these term by document matrices that we can construct through counting. Uh, we talked about one of the two transformations that we could do to those by applying term frequency, inverse document frequency. The other in the textbook is PPMI. Um, and we also talked about how you could move from that idea that was developed in information retrieval, which is really, uh, really conceptualized the value of those matrices as a way of retrieving similar documents.	The term by document matrix is a way of retrieving similar documents.
So if your query was thought of as a document, you could pull related queries or sorry, related documents to that query by querying the co-signed similarity for all the documents in your term by document matrix. Um, and then we saw how you could extend that term by document matrix idea to get word semantics by having a term by context matrix or a term by term matrix. Um, so those term by term matrixes are parameterized in a lot of different ways. We could think about how many words of context we want to take into account.	We can query a term by document matrix for related documents by querying the co-signed similarity for all the documents in the matrix. We could also think about how many words of context to take into account when computing semantic similarity.
We could think about, um, adding interesting linguistic context. Like we saw through the dirt method where they had dependency information. So instead of immediately adjacent words, they looked at, uh, parent of child of grandparent, of grandchild of et cetera, but they all kind of had the property that the vectors that resulted from these various methods for, uh, creating the term by context matrix meant that the representation of words were long and sparse the length of the vectors tended to be some function of the vocabulary size.	We could think about adding interesting linguistic context to our word representations by looking at words that are higher up in the dependency tree. The length of the vectors tended to be some function of the vocabulary size.
Um, and the sparsity results in effect that by looking at a, a sliding window around a word, you're not gonna encounter that word. Co-occurring with all words in the vocabulary of English. So there's gonna be lots of zeros. So the, the place that we pivoted to at the end of last lecture was to start looking at the more modern, um, representation that we use for words still in the same vector space idea, called word embeddings, where the major difference is the vectors themselves.	The modern representation for words is called word embeddings. The major difference is that the vectors themselves are learned through training.
Instead of being the length, the size of the vocabulary are gonna be much, much shorter. We're going to be able to specify however many dimensions we want to use to encode the, the, uh, representation of the word. And usually we pick some relatively small number of dimensions, like on the order of 100 or 300.	We can specify as many dimensions as we want for the representation of a word. Usually we pick some small number, like 100 or 300.
um, and partially as a result of picking a much smaller number and as a result of how we are then going to train the values, uh, to be included in the representation of each word, the vectors then move from being sparse vectors with lots of zeros to dense vectors, where we have almost no zeros. So the algorithm that we briefly looked at last time, um, uh, was called word Tove, which produces these dense vectors.	The algorithm that is used to generate word embeddings is called word2vec. The vectors that are produced are dense vectors, which is different from the sparse vectors that we have seen before.
So the value of these dense vectors versus the sparse vectors, um, and the value of them be having a relative few number of dimensions is that they're much easier, uh, to use for things like machine learning. So for instance, if you were to train your classifier to say, is this word simple or complex, you could actually use those a hundred dimensions as features in your classifier.	The value of dense vectors for machine learning is that they are much easier to use than sparse vectors.

Table 3: Example summaries generated by our fine-tuned summarizer

Lexical Semantic Post-Quiz Question

1. Vector space semantics is a representation of word meaning. (T/F)
2. In N-gram language models, words are represented as
   (a) Vectors
   (b) Tokens
   (c) Scalars
   (d) Lists of ASCII codes
3. In the N-gram model, the following words (introduce, introducing, introduced) are treated as the same token. (T/F)
4. What are the drawbacks of simply using N-gram models?
   (a) We really didn't understand that there was a relationship between those different variants of the same underlying word
   (b) We can run into the problem of sparse counts
   (c) Both of A and B
   (d) None of A and B
5. When we encode word meanings, we should consider the property that words can be synonyms, meaning that they have similar meanings to other words that are totally different. (T/F)
6. What is the LEAST important factor that we should consider while encoding word meanings?
   (a) Words have synonyms
   (b) Words that have opposite meanings
   (c) Words that reflect different semantic roles
   (d) The number of word occurrences
7. Semantic representation will not involve encompassing syntactic positions of the words. (T/F)
8. WordNet is an example of ___ knowledge base.
   (a) Syntactical
   (b) Lexical
   (c) Grammatical
   (d) Pronunciation
9. WordNet does not encode hierarchical organization of words. (T/F)
10. Which of the following refers to words that are more general than the current word?
    (a) Hypernym
    (b) Hyponym
    (c) Synonym
    (d) Antonym

Stochastic Gradient Descent Pre-Quiz Questions

1. At each step, gradient descent finds out the direction along which the function changes the most quickly, and moves in this direction. (T/F)
2. In gradient descent, at each step, what should we know to update the weight?
   (a) Previous weight, learning rate, slope value
   (b) Previous weight, learning rate, previous function value
   (c) Previous weight, previous function value
   (d) Learning rate, slope value
3. What is the role of learning rate in gradient descent?
   (a) To decrease the weights and avoid very large weights
   (b) To control the step size of our move in gradient descent at each step
   (c) To learn from the training set
   (d) To account for overfitting
4. Gradient descent can not be used for weights with multiple features. (T/F)

Stochastic Gradient Descent Post-Quiz Questions

1. At each step, gradient descent finds out the direction along which the function changes the most quickly, and moves in this direction. (T/F)
2. In gradient descent, at each step, what should we know to update the weight?
   (a) Previous weight, learning rate, slope value

(b) Previous weight, learning rate, previous function value
(c) Previous weight, previous function value
(d) Learning rate, slope value

3. What is the role of learning rate in gradient descent?

   (a) To decrease the weights and avoid very large weights
   (b) To control the step size of our move in gradient descent at each step
   (c) To learn from the training set
   (d) To account for overfitting

4. What might be the problem when our learning rate is too big for convex functions?

   (a) We will have very large weights at the end
   (b) We will have very small weights at the end
   (c) We will move back and forth in gradient descent update and never find the global minimum
   (d) There is no problem with a very large learning rate

5. For logistic regression, gradient descent can always find the global minimum of its loss function. (T/F)

6. Gradient descent can not be used for weights with multiple features. (T/F)

7. For convex functions, gradient descent with a reasonable learning rate can always find the global minimum. (T/F)

8. For convex functions, the starting point where we start gradient descent is not important. (T/F)

9. Gradient descent is a method that uses which of the following to determine the minimum of a function

   (a) The function's current value
   (b) The function's intercept
   (c) The function's slope
   (d) The function's maximum value

10. Gradient Descent is guaranteed to find the minimum of the logistic regression loss function because

   (a) We use very powerful machines to run the method
   (b) The loss function is convex
   (c) The loss function is concave
   (d) We start gradient descent from a carefully chosen point

E Summaries From Our User Study

E.1 Lexical Semantic Lecture Summary

0:29 - The current most exciting trendy topic in NLP is how to represent the meaning of words. This will be discussed through a particular style of representation called vector space semantics.
0:29 - The history of vector space semantics goes back to the information retrieval systems. More recently, we have changed the way we create vectors by using neural networks.
0:29 - The problem with traditional language models is that they do not understand the relationship between words. In this class, we will discuss ways to create neural language models that do have this understanding.
1:38 - The problem with N-gram based language models is that we can run into the problem of sparse counts, where we cannot see how likely it is that some other variants of a word will appear at test time if we only have a small sample of data from which to learn.
2:39 - There are many different elements that we would like a good representation of word meaning to be able to encode. One is the idea that words can be synonyms even though they are totally different in terms of their spelling.
2:51 - We will discuss how to measure similarity between words, as well as how to understand the opposite and similar meanings of words.
3:36 - There can be words that reflect different semantic roles, and words that have a positive or negative connotation.
3:25 - Entailment is an important aspect of word meaning.
6:48 - Entailment can be mapped onto language in a way that reflects the meaning of words.
8:47 - Entailment can be used to make inferences. For example, if we know that all animals have an old and their artery, then we can infer that dogs must have an old and their artery.
4:51 - The ability to use logic as a representation of the meaning of language would give us a very powerful machinery for handling inferences and entailments.

8:46 - The downside of using formal logic as a representation for the meaning of language is that we have to acquire that knowledge. There are, however, resources that have been created that help with this problem. One very important resource is called WordNet, which is a lexical knowledge base containing the meaning of words.
7:23 - WordNet has synonym sets that represent different senses of a word. Each sense of the word is represented as a distinct concept.
8:59 - The WordNet resource encodes the meaning of words in a way that reflects the hierarchical organization of words in the language.
8:47 - In WordNet, a dog is a kind of canine and a domesticated animal. Clicking on each of these concepts shows how they are related through inheritance.
8:46 - In order to make an entailment, we need to be able to walk through the different levels of hierarchy in WordNet.

E.2 Stochastic Gradient Descent Lecture Summary

0:00 - We want to find a parameter setting that minimizes this loss over all of the items in our training set. Theta is the set of parameters that we have at our model.
0:54 - We want a good setting of theta that minimizes the average loss across our training set using the cross entropy loss function.
1:52 - The algorithm that is used to find the optimal parameter setting is called gradient descent.
1:36 - The algorithm for finding the optimal parameter setting is called gradient descent. The algorithm uses a method for pushing around values in a weight vector to find the optimal setting.
2:23 - The algorithm uses a method for pushing around values in a weight vector to find the optimal setting. The analogy for thinking about this is you're in a canyon and you want to find your way down to the river.
2:51 - The idea of a function and what its minimum point is can be used to understand the idea of gradient descent.
3:20 - The loss function for logistic regression is convex, which means there's just one minimum for logistic regression. As a result, gradient descent starting from any point is guaranteed to find the minimum.
4:00 - The loss function for logistic regression looks like this. We can decrease w by pushing it in this direction and we can increase w by pushing it in this direction.
4:22 - We want to find the point where the loss is the lowest by computing the slope of w with respect to our loss function.
2:00 - We will compute the slope of w with respect to our loss function and take one step in the direction of the slope.
6:53 - The learning rate is the amount by which we step in the direction of the slope.
1:36 - The weight at each time step is the current weight at the previous time step minus the learning rate, which is the step size. The slope is the derivative of the loss function with respect to the weight, and then we add back in the learning rate.
7:22 - The reason the curve goes up after we cross the minimum is because this is just how we drew this particular loss function. The minimum is always going to be with respect to the loss.
9:20 - The minimum point is the best weight associated with one of our features.
8:30 - This is a convex optimization problem, which means there's only one minimum. Other types of problems are nonconvex, which means there can be multiple minimums.
9:90 - The idea of taking a step in the direction of the slope may not work for nonconvex problems, which are problems with multiple minimums.
9:43 - The learning rate is how much we should step in the direction of the slope. There's interesting literature on how to set the learning rate for nonconvex problems.
10:43 - We want to use the intuition of moving left and right for a single value to move left and right for multiple variables.
11:14 - We will take the gradient of the weight across many dimensions and use that to find the minimum.
11:14 - We will use the intuition of moving left and right for a single value to move left and right for multiple variables.
11:25 - The intuition is that moving left and right for a single value should move left and right for multiple variables.

F Student performance on each quiz

From the pre-quiz and post-quiz results, as shown in Tables 4 and 5, we can calculate the increase in percentages of mean correctness of the different conditions for Lexical Semantic lecture. Refer to

Table 6 for this analysis. Similarly, we calculate the increase for the Gradient Descent lecture, as shown in Table 7. Note that values are normalized when calculating the percentage increase.

Condition	Mean Correctness	Std Dev
Video	2.8	0.94
Summary	3.24	1.03
V+S	2.88	0.93

Table 4: Pre-Quiz results Lexical Semantic

Condition	Mean Correctness	Std Dev
Video	7.92	1.62
Summary	7.72	1.37
V+S	8.33	1.25

Table 5: Post-Quiz results Lexical Semantic

Condition	% increase
Video	13.14%
Summary	-4.69%
V+S	15.69%

Table 6: Percentage increase from Pre-Quiz to Post-Quiz results in Lexical Semantic Lecture

Condition	% increase
Video	26.48%
Summary	21.75%
V+S	35.33%

Table 7: Percentage increase from Pre-Quiz to Post-Quiz results in Stochastic Gradient Descent Lecture