**A Bit of Progress and Stronger-gram Language Modeling Baselines**

Summary

The document discusses the advantages and performance of n-gram language models compared to neural language models. It explains various smoothing techniques for n-gram models and presents experimental results demonstrating their competitive performance, especially in resource-lean setups and morphologically rich languages with high out-of-vocabulary rates. While neural models outperform n-gram models in data-intensive scenarios, the document suggests further research on the correlation between training corpus size and the effectiveness of these models. It concludes by highlighting the cost-effectiveness and competitive performance of n-gram models in challenging language settings and suggests exploring heuristics and continuous space representations to bridge the gap between n-gram and neural models.

N-gram model advantages.

This section of the document discusses the progress and improvements in n-gram language modeling compared to neural language models. It highlights the performance of n-gram models in various experiments and languages, showing that they can outperform neural models in certain scenarios. The section also discusses the advantages of n-gram models, such as their ability to handle out-of-vocabulary words and rare words without pruning. It provides an overview of established smoothing techniques for n-gram models, focusing on the top-level interpolation. The Kneser-Ney and Modified Kneser-Ney smoothing techniques are explained, along with their parameters.

Types of language models and their performance compared to neural LMs - Modified Kneser-Ney, Generalized Modified Kneser-Ney, Bayesian Kneser-Ney. Handles OOV words. Competes with neural LMs in resource-lean setups.

This section of the document discusses different types of language models (LMs) and their performance compared to neural LMs. It introduces the Modified Kneser-Ney (MKN) model, the Generalized Modified Kneser-Ney (GKN) model, and the Bayesian Kneser-Ney (BKN) model. The document also explains how these models handle out-of-vocabulary (OOV) words. The section includes tables showing the perplexity scores of different LMs for various languages and discusses their performance in challenging resource-lean setups. Overall, the n-gram models are shown to be highly competitive with neural LMs in these setups.

N-gram > neural models

This section of the document discusses the comparison between n-gram language models and neural language models in the context of morphologically rich languages with high out-of-vocabulary (OOV) rates. The section presents experimental results that show that n-gram language models, especially the BKN variant, outperform neural models in these language settings. The section also highlights the importance of using more sophisticated n-gram models like BKN or GKN instead of the commonly used KN or MKN models for adequate comparison.

Additionally, the section mentions that while neural models outperform n-gram models in data-intensive scenarios, training on large datasets comes at the expense of training efficiency. The section suggests further research on the correlation between training corpus size and the effectiveness of n-gram vs. neural language models.

Lastly, the section concludes by emphasizing the cost-effectiveness and competitive performance of n-gram language models compared to neural models, especially in challenging language settings. It suggests exploring cheap-to-compute heuristics and utilizing continuous space representations to bridge the gap between n-gram and neural models.

#### Beyond N-Grams- Can Linguistic Sophistication Improve Language Modeling?

Summary

The document discusses the limitations of current language models for speech recognition and explores the use of linguistic knowledge to improve these models. Experimental results show that people are able to improve speech recognition accuracy by using linguistic knowledge, particularly in closed class word choice and world knowledge. The section suggests that further improvements may require solving linguistic problems and mentions areas for future work, including expanding the study and using the findings to enhance speech recognition.

Improving speech recognition.

This section discusses the limitations of current language models for speech recognition, which are based on n-grams, and explores the possibility of improving these models by incorporating more sophisticated linguistic and world knowledge. The section also describes previous attempts to incorporate linguistic information into language models and the challenges in achieving significant improvements. The section then presents an experimental framework to gain insights into what linguistic knowledge could contribute to improving language models for speech recognition. The experiments involve human subjects attempting to improve the output of speech recognition systems and recording the types of knowledge they used in doing so. The results show that people are able to improve the speech recognizer's output on various corpora, with greater improvements seen as the accuracy of the recognizer increases. The section also highlights the usefulness of linguistic information in improving speech recognition and the importance of considering methods for editing rather than solely choosing from the highest-ranked hypotheses. Overall, the section emphasizes the potential for linguistic sophistication to enhance language models for speech recognition.

Study on speech recognition accuracy using knowledge types: closed class word choice, world knowledge, linguistic proficiencies. Potential for algorithmic replication. Future work: larger study, refined analysis, understanding human usage, improving recognition.

This section of the document discusses the results of a study on the use of different types of knowledge to improve the accuracy of a speech recognizer. The study looked at three different corpora and analyzed the percentage of instances where people used different types of knowledge and the net gain in accuracy achieved by using that knowledge.

The most common and useful proficiency that people used was closed class word choice, which involved distinguishing between similar words such as "in" and "and" or "than" and "that". World knowledge was also frequently used, but there were many linguistic proficiencies that contributed to the improvement in accuracy.

The section concludes by stating that while there is still a lot to gain from postprocessing the recognizer's output, further improvements may require solving linguistic problems. However, algorithms could potentially mimic many of the proficiencies used by humans.

The section also mentions four areas for future work: expanding the study to include more people, conducting a more refined study at a lower level of granularity, determining how humans use the proficiencies, and using the findings to improve speech recognition.

**Building Wikipedia N-grams with Apache Spark**

Summary

The document discusses the construction of N-grams for language modeling using Apache Spark. It explains the distributed data processing approach for building N-gram corpus and describes the Apache Spark architecture and cluster setup. An experiment is conducted using Spark to build an N-gram corpus from an English Wikipedia archive file, and the results show significant reduction in runtime compared to single machine solutions. The section also provides a list of references for further study and implementation of building N-grams with Spark.

N-grams in Spark

This section of the document discusses the construction of N-grams for language modeling using Apache Spark. N-grams are defined as a continuous sequence of words of length N and are useful in language models. The document introduces a distributed data processing approach to build an N-gram corpus, which is faster compared to previous single machine techniques. The section also discusses the related works in generating N-grams and the limitations of existing resources. It then explains the Apache Spark architecture for data processing, including the concept of Resilient Distributed Datasets (RDD) and the transformations and actions available on RDD. The section concludes by describing the Apache Spark cluster architecture with a driver application node and worker nodes.

Spark N-gram experiment.

This section of the document describes an experiment conducted using the Spark framework to build an N-gram corpus from an English Wikipedia archive file. The dataset used in the experiment is a compressed XML file with a size of 78 GB. The experiment involves multiple steps, including converting the XML file to a binary file format called Parquet, performing data cleaning transformations, and extracting N-grams from the cleaned data. The cluster setup used for the experiment consists of 10 worker nodes. The section also provides details on the data processing steps, including the use of the Spark XML library, data cleaning using regular expressions, and N-gram extraction using the N-gram library from the Spark ML package. The results of the experiment show that the total runtime for generating N-gram tokens of size 1 to 5 is 58.7 minutes using the Spark SQL API, compared to 35 hours for a single machine solution. The section concludes by mentioning future studies that will use the Spark framework to extract more features from text documents and improve data processing and training time complexity.

N-gram references for Spark

This section of the document contains a list of references related to building n-grams with Spark. Some relevant details include:

- The references include papers and articles on topics such as text classification using neural network language model (NNLM) and BERT, sentiment analysis for social media bot detection, machine learning algorithms for social media bot detection, and offensive language detection on social media based on text classification.

- There are also references to resources such as the Apache Spark documentation, the Apache Parquet library, the Hadoop distributed file system, and the MavenRepository for the Databricks Spark XML Package.

- Additionally, there are references to papers on other topics such as clinical text classification of Alzheimer's drugs' mechanism of action, developing web-based machine learning applications, MapReduce for large-scale data processing, and resilient distributed datasets for in-memory cluster computing.

Overall, this section provides a list of resources and references that can be used for further study and implementation of building n-grams with Spark.