May 31, 2019

Dialog-aware language models for speech recognition

Benjamin Haynor

Petar Aleksic

Dialog-aware language models for speech recognition

ABSTRACT

Automatic speech recognizers (ASR) typically treat each utterance of a conversation independently. This often leads to errors such as the incorrect transcription of homophones. These errors cascade into further problems when performing natural language understanding. This disclosure presents speech recognition techniques that transcribe speech using the larger context of the dialog. Per the techniques, individual utterances are transcribed based on the context of the conversation. The techniques distinguish homophones by context and improve in-dialog ASR without relying on supervised data or manually-provided phrases. The techniques generalize well to unseen dialogs or queries.

KEYWORDS

- Automatic speech recognition (ASR)
- Language model
- Dialog-aware ASR
- N-gram
- Homophone
- Text-to-speech
- Natural language generator
- Non-linguistic token
- Spoken query
- Virtual assistant

BACKGROUND

Automatic speech recognizers (ASR) typically treat each utterance of a conversation independently. This often leads to errors such as the incorrect transcription of homophones, e.g., transcribing “c” as “see,” “die” as “dye,” etc.). These errors cascade into further problems when performing natural language understanding. With the rising popularity of voice-based digital assistants, the problem of accurately transcribing utterances in a dialog with a conversational agent has become increasingly important.

The cause of homophone errors as described above can be traced to the evaluation of probabilities by the language model. For example, in a conversation that unfolds thus:

```
Computer agent: Choose between a, b, or c
Human user: c,
```

the language model determines the probabilities P(“c” |<S>) and P(“see” |<S>), where <S> is the utterance within which the c-sound is detected. The language model determines that

$$P(\text{“see”}|\langle S\rangle) > P(\text{“c”}|\langle S\rangle),$$

and hence outputs “see” as the detected sound.

Composition-based on-the-fly rescoring for salient n-gram biasing is one current approach to accurate speech transcription [1]. Under this approach, clients provide dialog context manually. For example, in the multiple-choice conversation above, the ASR client sends a request context comprising the strings {“a”, “b”, “c”} to bias the ASR to the multiple choice answers. This approach suffers from several problems:

- It requires manual work on behalf of the developer to decide what phrases to bias towards.

- The approach cannot learn how strongly to bias towards the client-provided n-grams (for instance to bias more strongly towards {blue, green yellow} than {sarcoline, coquelicot, smaragdine} when seeking a response to a user's favorite color).
- Developers can easily miss a particular phrasing of a response. For example, a developer might bias towards {"yes", "no"} when asking a yes/no question, but forget about {"naw", "nope", "yeah"} etc.

Unsupervised context learning [2] is another approach that attempts to automatically learn the relevant n-grams to bias towards for a given context. In theory, this approach fixes the problems with the previous approach, but it introduces two of its own problems:

- It requires production data to learn how to bias the language model. This entails launching with a relatively untrained product, collecting data, and training a model. A disadvantage is that bugs that arise subsequently are difficult to detect and fix.
- The approach can actually bias towards errors. If the ASR system consistently makes an error, biasing often only makes that error worse. In the multiple choice example above, this approach may incorrectly learn to consistently bias towards "see".

DESCRIPTION

Per the techniques of this disclosure, a language model evaluates probabilities conditioned not on the immediate utterance (or query) but on the dialog history that includes the immediate utterance. In the example multiple-choice dialog, the language model evaluates the probabilities

P("see" | entire dialog history including the question the user asked) and

P("c" | entire dialog history including the question the user asked),

and outputs the sound with greater probability.

The language model is trained on dialogs including both user and computer agent turns. For example the dialog below is encoded in its entirety as training data, with conversational turns being encoded as special symbols in the language model.

```
User: Let's play a guessing game
Computer: Okay choose a, b, or c
User: c
```

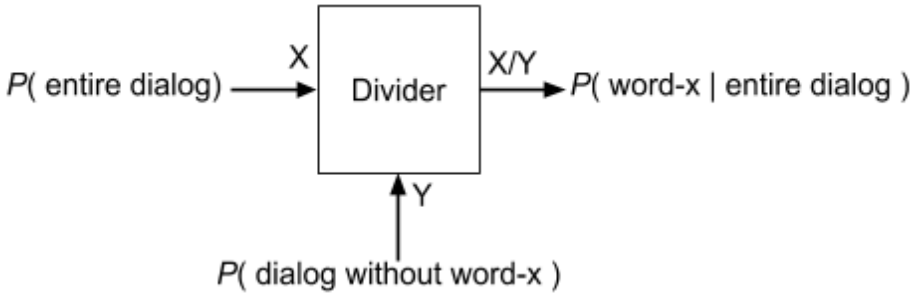

Fig. 1: Calculating the probability of an unknown word (word-x) given the entire dialog

Fig. 1 illustrates the recognition of a word (word-x) or phrase given the entire dialog, based on the principles of conditional probability. For example, in the above multiple-choice example (where word-x is "c"), to compute *P*(word-x | entire dialog), e.g.,

P("c" | "<USER> Let's play a guessing game <COMPUTER> Okay choose a, b, or c <USER>"),

the techniques divide *P*(entire dialog), e.g., the probability

P("<USER> Let's play a guessing game <COMPUTER> Okay choose a, b, or c <USER> c")

by the probability *P*(dialog without word-x), e.g.,

P("<USER> Let's play a guessing game <COMPUTER> Okay choose a, b, or c <USER>"),

which the language model provides.

The techniques have a number of features to cover variations in language model training and use, including the following.

Sources of training data

Sources of training data include typed dialog with a virtual assistant application, transcribed voice dialog with a virtual assistant application, unsupervised voice dialog with virtual assistant application, etc.

Representing computer agent turns by tokens rather than, or in addition to, strings

The modeling power expended on the computer side of the dialog can be saved by replacing the computer's turn by a single token. For example, the two conversations

```
Computer: Would you like to send it?
User: Yes,
```

and

```
Computer: Would you me like to send it?
User: Yes,
```

are both modeled as

"<COMPUTER> <GET_SEND_CONFIRMATION> <USER> yes".

The reduction in complexity brought about by representing the computer's turn by tokens leads to improved model performance and generalization.

The tokens can be incorporated in various ways, as follows.

- **Tokens can be used to replace strings:** When a text-to-speech ID is available, that ID is used instead of the computer's side of the conversational string.
- **Tokens can be used in addition to strings:** In order to generalize to unseen dialogs (where no IDs are available to the model), multiple training examples are generated with and without text IDs. Alternatively, multiple language models with text examples and/or tokens are generated to enable generalization to new dialogs with unseen text IDs.

- **Both text and text IDs are included in the same conversational string:** For example, rather than just "<COMPUTER> <GET_SEND_CONFIRMATION>", a conversational string can be "<COMPUTER> Would you like to send it <GET_SEND_CONFIRMATION>".

The tokens can be generated using various sources, as follows.

- **Text to speech (TTS) IDs:** Natural language generators (NLG) can generate various textual formulations of the same semantic turn by the computer agent. For example, given two strings "would you like to send it" and "would you like me to send it", an NLG generates the same ID.
- **IDs generated by conversational agents:** Conversation agents that develop dialog can generate IDs such as <PROMPT_FOR_TIME>, <PROMPT_FOR_DATE>, etc. These ids can be used in place of textual queries in training data for language models, e.g., "what time should I set the calendar event", "what day should I set the calendar event", etc.

Incorporating non-linguistic tokens into the language model

Non-linguistic information can be incorporated advantageously to the language model. Some types of non-linguistic tokens include:

- **Time-delay tokens:** In a long delay between queries, a time delay token is inserted into language model training data using quantized buckets. An example is the token <4_TO_8_HOURS_PASS> in the following pattern.

```
User: Good night.
Computer: Good night, turning off your smart lights.
<4_TO_8_HOURS_PASS>
```

```
User: Good Morning.
```

The bucketing strategy for tokens, e.g., <less-than-one-minute-pass>, etc. can be determined by experiment. A neural language model can possibly handle some continuous representations of the time interval.

- **Action tokens:** An example of an action token, <START_MUSIC> occurs in the following pattern, which can be used as training data.

```
User: Play some music.
Computer: Playing some jazz.
<START_MUSIC>
User: Pause music.
```

Other examples of non-linguistic tokens include, e.g., <ALARM_STARTS_FIRING>, <TIMER_SET>, etc. An advantage of this formulation is that it easily incorporates non-linguistic information into the sequence model as well.

Options for language modeling technique

Options for language modeling include n-gram language model, neural language model, etc. Under a neural language model, long-range dependencies can be modeled. For example, in the dialog

```
A: Would you like an apple or a banana?
B: I would like an apple
```

the distance between the two occurrences of “apple” is nine words, which is within the dependency-modeling abilities of a neural model. Further, the neural language model can generalize a rule, e.g., about the probability *P*(word-x | word-x previously in dialog)", independent of word-x.

Inference options

- For n-gram language models, the conversational language model can be implemented by an alternative start state. For example, given a dialog "<COMPUTER> Would you like an apple or a banana <USER>", rather than starting at the <S> (start) state, arcs are followed through the language model in order to respond to the question posed in the dialog.
- For neural language models with traditional ASR, this can be used as a rescoring language model.
- For exchange-to-exchange scenarios, the language model can be fused in a variety of ways.

Alternative to a single model that handles dialog context and non-linguistic contextual events, each dialog state can be treated differently, e.g., one language model can be trained for multiple-choice questions, another for yes-no type questions, etc. Further, language models can be interpolated based on dialog history by building several separate language models.

In this manner, the techniques of this disclosure improve speech recognition without relying on supervised data or manually-provided phrases. The techniques also have the ability to encode non-linguistic tokens, e.g., <ALARM_STARTS_FIRING>, <TIMER_SET>, etc. to improve recognition of queries like "snooze alarm", "cancel timer", etc. The techniques generalize to unseen queries by modeling the dialog turns of the computer as text or IDs.

CONCLUSION

This disclosure presents speech recognition techniques that transcribe speech using the larger context of the dialog. Per the techniques, individual utterances are transcribed based on the context of the conversation. The techniques distinguish homophones by context and improve in-dialog ASR without relying on supervised data or manually-provided phrases. The techniques generalize well to unseen dialogs or queries.

REFERENCES

[1] Hall, Keith, Eunjoon Cho, Cyril Allauzen, Francoise Beaufays, Noah Coccaro, Kaisuke Nakajima, Michael Riley, Brian Roark, David Rybach, and Linda Zhang. "Composition-based on-the-fly rescoring for salient n-gram biasing." (2015) available online at https://ai.google/research/pubs/pub43816.pdf

[2] Michaely, Assaf Hurwitz, Mohammadreza Ghodsi, Zelin Wu, Justin Scheiner, and Petar Aleksic. "Unsupervised context learning for speech recognition." In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 447-453. IEEE, 2016.