Scikit-talk: A toolkit for processing real-world conversational speech data

Andreas Liesenfeld, Gábor Parti and Chu-Ren Huang
The Hong Kong Polytechnic University
Hong Kong SAR, China
amliese@polyu.edu.hk, gabor.parti@connect.polyu.hk,
churen.huang@polyu.edu.hk

Abstract

We present Scikit-talk, an open-source toolkit for processing collections of real-world conversational speech in Python. First of its kind, the toolkit equips those interested in studying or modeling conversations with an easy-to-use interface to build and explore large collections of transcriptions and annotations of talk-in-interaction. Designed for applications in speech processing and Conversational AI, Scikit-talk provides tools to custom-build datasets for tasks such as intent prototyping, dialog flow testing, and conversation design. Its *preprocessor* module comes with several pre-built interfaces for common transcription formats, which aim to make working across multiple data sources more accessible. The *explorer* module provides a collection of tools to explore and analyse this data type via string matching and unsupervised machine learning techniques. Scikit-talk serves as a platform to collect and connect different transcription formats and representations of talk, enabling the user to quickly build multilingual datasets of varying detail and granularity. Thus, the toolkit aims to make working with authentic conversational speech data in Python more accessible and to provide the user with comprehensive options to work with representations of talk in appropriate detail for any downstream task. For the latest updates and information on currently supported languages and language resources, please refer to: https://pypi.org/project/scikit-talk/

1 Introduction

Real-world conversational speech data, also known in the designated fields of study as transcription of talk-in-interaction, is complex. Talk can be transcribed in varying level of detail and symbolic representations of elements in talk range from simple word-level transcripts to fine-grained phonetic representations and multi-layered xml tags for all sorts of audio-visual information that the representation aims to capture. When using this data type for any downstream task, an important step is to decide on the appropriate level of granularity and to strive for a systematic representation format. Scikit-talk is designed to aid anyone interested in processing talk with these decisions and to make building datasets of talk more efficient, flexible and accessible.

Designed for applications in speech processing and Conversational AI, Scikit-talk provides tools to custom-build datasets for tasks such as intent prototyping, dialog flow testing, and conversation design. An important step in the development of commercial task-oriented bots is the initial design of intent labels and dialog flows. Depending on the task, prototyping intent flows can be difficult. Breaking down the interaction of ordering a pizza into discrete intents such as "select pizza type" and "input delivery address" is straight-forward. But, for instance, designing dialog flows for more complex interactions, such as bots for technical support tasks, help, or FAQ hotlines can be more complex. How will these interactions unfold? What are typical dialog flows for such interactions in the real world? Scikit-talk provides an accessible tool for tasks such as bot prototyping for Conversational AI. It offers a unified toolkit to process and query databases of real-world interactions to find and analyse instances of a certain type of interaction for data-driven conversation design and development cycles (Figure 1).

Prototype interaction flows sometimes do not match well with how users actually complete the task in real-world scenarios, which can result in additional development cycles until a satisfactory performance is reached (Yang et al., 2019). Initial design decisions may require revision later on, such as editing, merging or deprecating intents once real users interact with the bot. Scikit-talk addresses this by providing builders and designers with a tool

Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue, pages 252–256
July 29–31, 2021.

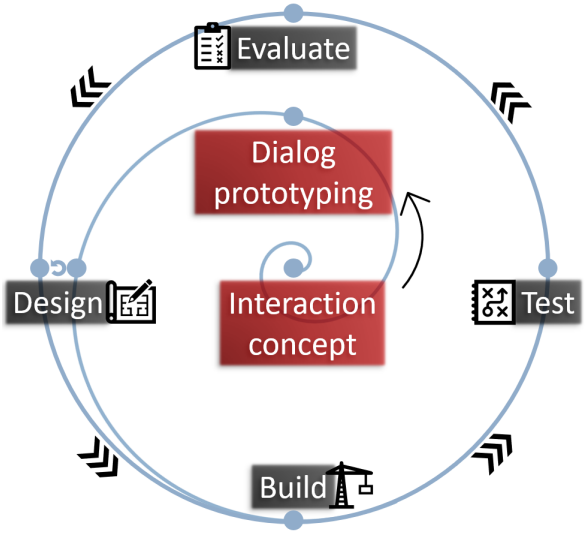

Figure 1: A typical bot development spiral: from prototype design to development and deployment cycles

to explore how to better match intent labels and dialog flows in the prototype stage. For instance, when prototyping of a bot for a customer support task, Scikit-talk can be used to explore similar explanatory sequences in real-world data. This helps builders to ground their prototype in authentic data and mitigate the risk of grounding development cycles on artificial premises.

Scikit-talk is the first open-source tool for processing transcripts of authentic speech specifically designed for research and development. It comes with a range of data exploration tools for collocation and concordance analysis, as well as a pipeline to automate annotation, analysis, and visualization of interaction types or specific NLU intents (such as greeting, accusing, deflecting, or apologizing).

In contrast to prototyping based on typed chat or movie subtitle data, Scikit-talk enables insights that take more detail of real talk into account, including seemingly incoherent or messy discourse structures, disfluency, repair, repetition or restarts and other features that are common in natural spoken interaction. Designing for real talk means designing closer to real-world scenarios.

Figure 2 shows an example of intent label and dialogue state design grounded in conversational data. The analysis highlights some of the issues that arise when working with this data type, such as labelling various turns and dealing with features of conversational interaction such as abandoned turns or joint turn completions.

Inspired by Scikit-learn, the toolkit consists of several modules to preprocess, explore and analyse this data type that were designed with explainability and customisation in mind (Pedregosa et al., 2011).

The toolkit also enables the user to combine existing language resources using pre-built interfaces for common corpus transcription formats. The motivation behind Scikit-talk is to build a tool to conveniently explore large collections of authentic talk. This aim is reflected in the following design decisions:

- **The fundamental unit of organization is the turn, not the sentence.** A turn is an utterance of any length by a speakers that only ends when another speakers begins to talk. This means that data is segmented only by speaker change, not punctuation.
- **Compile datasets using custom transcription formats as well as existing language resources** Scikit-talk should provide the option to configure custom formats as well as maintain a collection of pre-built interfaces for popular existing language resources of conversational speech.
- **Retain as much detail of authentic speech as the transcription formats permit.** No repetitions, repairs, or non-lexical utterances are discarded. This often includes any available transcriptions of utterances such as “oh” or “erm”, laughing, sighing, pauses, truncation and overlapping speech.
- **Utterances should be explored within their interactional contexts.** Scikit-talk should provide a collection of state-of-the-art tools for concordance and collocation analysis of lexical as well as non-lexical elements in talk.
- **Make the tool available to anyone interested in processing real-world conversational speech data**. For the given tasks, the toolkit should lower the technical barrier of access and adoption, making it a useful open-source platform for developers, researchers and designers alike.

2 Overview of the Scikit-talk toolkit

The idea of a toolkit for conversational speech processing is inspired by an existing similar tool for online chat conversations (Chang et al., 2020). Scikit-talk aims to provide both a unified framework for several existing transcription conventions as well as

a range of tools to explore this data type in Python. The *Preprocessor* module is used to combine several existing datasets via pre-built interfaces for various data formats while preserving as much transcribed information as possible. Apart from corpus building, Scikit-talk also includes tools for data manipulations. It includes string matching tools to explore and identify features of interest. And it includes tools for unsupervised machine learning techniques to explore distributional patterns in the data.

As for custom datasets, Scikit-talk can be used to analyse any transcriptions of talk in a turn-based format where the conversation unfolds in the "ABAB..." pattern for two speakers or "ABCBCA..." for three speakers, etc.[1]

3 Pre-built corpus interfaces

The main challenge of providing a comprehensive tool to explore how people talk in natural settings distributionally is that language resources of natural conversational speech are few and far between. Unlike movie subtitle corpora, these datasets are small because building them requires painstaking manual transcription and annotation. Existing large-scale conversational corpora often come with a custom annotation formats. While this may be for good linguistic reasons, it poses a significant barrier to facilitate work across different datasets. Processing and finding instances of a specific type of interaction or task in these datasets is hard because transcription conventions and data representation formats vary. Addressing this challenge, Scikit-talk provides a platform that aims to collect and connect existing computer-readable transcription formats of talk.

This fragmentation of representation formats poses a challenge to create custom datasets from multiple data sources or across existing corpora. Scikit-talk here provides a practical solution by providing a module that converts data across several common formats, the CHILDES CHAT format[2], that of the Spoken British National Corpus (SpokenBNC)[3], and the UPenn Linguistic Data Consortium (LDC)[4]. This means the current version of the toolkit already covers several of the largest available datasets of authentic conversations, such as SpokenBNC or any LDC datasets. More corpus interfaces for major language resources in more languages are planned.

Preprocessor enables the user to interface data from different corpora which can significantly reduce the workload of building large datasets and unifying data formats. The challenge here is that transcription formats may differ in convention and granularity, some providing more detailed information than others. *Preprocessor* merges transcription features that are consistent across the formats while automatically discarding those that are not. This minimizes the loss of information for the sake of consistency. Typically, consistency issues arise regarding the transcription of non-lexical conduct, pauses, overlaps, restarts and phonetic reductions. The module has been tested with SpokenBNC, LDC and CHAT format data in British English and LDC and CHAT format data in Mandarin Chinese. Combining several datasets using *Preprocessor* results in a single, large dataset that enables the user to build collections of interactions that are more comprehensive, provide broader coverage of a phenomenon, and enable more methods of statistical analysis (Figure 3).

3.1 String matching tools

The first step of exploring interactions, intents, and dialog flows in Scikit-talk is usually to compile a set of keywords to query the custom dataset. This can be keywords related to individual actions such as apologies or words typically occurring in a certain type of interactions or talk on a certain topic. The *Explorer* module also comes with tools for basic corpus query such as frequency, concordance, and collocation analysis.

The Regular Expression-based *Annotator* tool allows the user to build a collection of instances of a type of interaction or intent. It can also be used to annotate linguistic features for a more detailed analysis to explore how, for instance, apologies are used across the dataset.

3.2 Unsupervised machine learning tools

The *Cluster analysis* module offers tools to conduct exploratory data analyses using various unsupervised machine learning techniques.

Specifically, the module provides access to a range of dimensionality-reduction (*Multidimensional scaling* and *t-SNE*) and clustering techniques (*Hierarchical Agglomerative Clustering*). These

[1] Or any other possible order of any number of speakers.
[2] https://talkbank.org/manuals/CHAT.pdf
[3] http://cass.lancs.ac.uk/cass-projects/spoken-bnc2014/
[4] https://www.ldc.upenn.edu/

Turn	Speaker	Corpus excerpt	Intent labels	Dialog state	Challenges
T_1	A:	yeah I was trying to get my guitar fixed actually in the I was trying to tune it and the tuner didn't seem to work properly so I took it	problem statement	Task initiation	
T_2	B:	[overlap] get a get at tuner on your app [pause] you can get a free one	initiate solution proposal	Solution proposal	
T_3	A:	[overlap] really?	display interest; request more information	Explain problem	
T_4	B:	yeah on the	positive response; abandoned turn		
T_5	A:	[overlap] when then no the one I've got you you attach it to the guitar	question need; provide problem details		
T_6	B:	mm	backchannel, continuer		
T_7	A:	and you pluck the string and it you know	(cont'd) provide problem details		
T_8	B:	yeah	continuer		
T_9	A:	[overlap] go			abandoned turn
T_{10}	B:	[overlap] this one you just erm	continue solution proposal		
T_{11}	A:	get it on your phone?			joint turn completion
T_{12}	B:	yeah [pause] er [pause] it's called Guitar Tuner just hit that [pause] yeah let me just go and get the guitar quickly [long pause] so I start [pause] just that's typical isn't it? do it's not working oh I know sorry [pause] do it again [long pause] it's [unclear]	positive response; accept joint turn completion; continue solution proposal; demonstrate solution		
T_{13}	A:	oh oh	display understanding; change of knowledge state		
T_{14}	B:	sounds a little bit up there yeah	continue solution proposal; provide example	Explain solution	
T_{15}	A:	so how would I get this app then?	accept solution; proceed to subtask; information-seeking question		
T_{16}	B:	you just search it on the whatever search thing you've got on your phone I don't know [pause] erm	response		
T_{17}	A:	but mine my phone's not an er er a smart phone right?	information-seeking question		
T_{18}	B:	but you've got apps on it	response		
T_{19}	A:	ye-			truncated utterance
T_{20}	B:	[overlap] must be able to download them	(cont'd) response		
T_{21}	A:	I think I can er yeah I'll I'll have a look when I'm at home cos that only works like I think when the Wi-Fi's on	accept response	Accept solution; Task completion	

Figure 2: Example of data-driven intent label and dialog state analysis

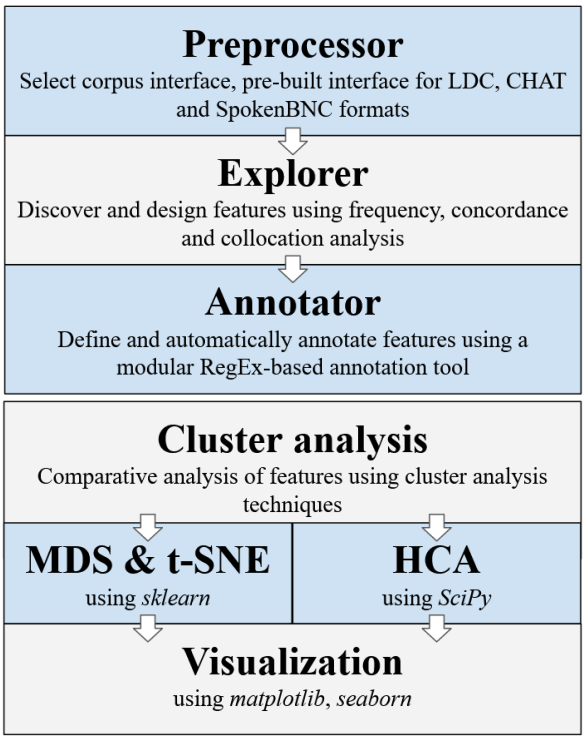

Figure 3: Scikit-talk module overview

tools enable further comparative analysis of linguistic patterns by comparing the (dis)similarity of any annotated features. The module also includes a range of customisable visualization tools.

Figure 3 shows a typical Scikit-talk workflow. First a dataset is defined. If the data is transcribed following one of the pre-built formats, the data can be interfaced in only one line of code. Based on a unified representation format, possible next steps are various string matching operations such as using the built-in concordancer to explore the dataset or annotating features for collocation analysis using various clustering techniques.

4 Conclusion and future work

Scikit-talk provides a platform to represent spoken conversational data, build datasets, and explore how certain types of interaction unfold in authentic talk. Tailored to the research and development community, the open-source toolkit makes working with transcriptions of talk across corpora more accessible for anyone interested in processing this data type. Scikit-talk features a *Preprocessor* tool that provides pre-built corpus interfaces for (currently three) popular transcription formats and enables rapid construction of unified data representations. The *Explorer*, *Annotator*, and *Cluster analysis* modules provide streamlined data exploration tools for collections of specific interaction or intent types using various string matching and unsupervised machine learning techniques.

In the future, Scikit-talk will be extended to provide additional corpus interfaces, covering more resources in more languages. Our aim is to maintain compatibility with future major releases of language resources of real-world conversational speech and to provide the research community with a useful tool for conversational speech processing to study and model talk in all its glori-

ous detail, grounded in the ways how people actually communicate. More information on the current state of this project can be found here: https://pypi.org/project/scikit-talk/

References

Jonathan P. Chang, Caleb Chiam, Liye Fu, Andrew Wang, Justine Zhang, and Cristian Danescu-Niculescu-Mizil. 2020. ConvoKit: A toolkit for the analysis of conversations. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 57–60, 1st virtual meeting. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *The Journal of machine Learning research*, 12:2825–2830.

Qian Yang, Justin Cranshaw, Saleema Amershi, Shamsi T. Iqbal, and Jaime Teevan. 2019. Sketching NLP: A Case Study of Exploring the Right Things To Design with Language Intelligence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–12, New York, NY, USA. Association for Computing Machinery.