DEVELOPING AUTOMATED TOOLS FOR POLITICAL FACT-CHECKING

MATTHEW O’ BOYLE¹, SHERRY FENG¹
¹Duke Department of Computer Science

INTRODUCTION
The current process of fact-checking is, in a sense, “chaotic”. In the current process, independent fact-checkers employ upwards of 20 fact-checkers to assess the authenticity of major claims made in a speech. Journalists then filter claims to assess what is ostensibly “checkworthy”, and manually search through an archive of fact-checks to identify suitable matches. If a match exists, commentary about the claim is written and a rating is published; if not, a reporter starts researching the claim, and then an editor revises and publishes it with a claim. This entire process can take a day at least, upwards of a week. Our team is automating the matching process in order to assist reporters who are otherwise manually seeking through the archives. This has the potential to improve efficiency by hours or even days, and moves us towards a more fair and transparent world for media.

OUR INTERFACE FOR DATA VERIFICATION
Our machine learning model, as is in its incipient stages, requires some human input to ensure the model is trained and accurate. We created a web interface in React and Ruby that allows journalists to match claims to fact-checks, and select fact-checks they want to present to the end user.

Clicking the time allows users to go back to that point in the video and verify the transcription

Keyboard shortcuts enable journalists to prioritize speed, crucial in live fact-checking

Green statements have already been manually verified by other journalists to be relevant to the claim

During observation, we found that time is the most valuable commodity since we have a limited number of fact-checkers.

Selecting the fact-check associates the claim as relevant to that listed fact-check

Our interface substantially reduces the work load on journalists. The interface could speed up the process by hours or even days, and moves us towards a more fair and transparent world for media.

Challenges
- Speed was the biggest challenge. To enable this, we only provide the headline from the fact-check article in our interface.
- However, this may not be enough information to make judgement calls, and providing a link to the article may result in lesser speed efficiency in the long term, since users would have to click to read.
- A secondary challenge was accuracy. When a user detects a mis-translation or mis-grouping, the updated transcription would have to run back through the pipeline, wasting valuable time in the process.

Design Challenges
What do we optimize for when designing this web interface? The biggest challenge here was thinking about how we could design an interface that would be fast and intuitive. The two main challenges:

- Speed: since journalists are fact-checking in real time, we prioritized speed over all else, adding keyboard shortcuts and only offering the headlines of fact-checks to compensate, acknowledging that this came at the tradeoff of information clarity.
- Concurrency: since multiple fact-checkers could be performing the same fact-check at the same time, we need some way of indicating conflicting edits. UI features to solve this are listed to the right.

Problems with Automation
- Mistranslation: Google’s speech-to-text “mishears” a sentence. This is heightened by applause and politicians talking over each other
- Incorrect Grouping: Our speech-to-text component tries to identify where sentences begin and end, but often splits check worthy sentences into two seemingly irrelevant parts
- Difficulties with context: Our algorithm often fails to determine what “it”/“that” is referring to when a politician has mentioned a topic earlier in their speech

Problems with Automated Tools
- Blue alert box highlights when other journalists are editing the stream; prevents concurrency conflicts
- Green statements have already been manually verified by other journalists to be relevant to the claim
- Gray boxes highlight when other journalists are editing, preventing conflicts

Future Goals
- Search Feature. To avoid transcription errors, we aim to allow users to input their own text to search for fact-checks if they feel the algorithm has mismatches or mistranslated.
- Curator Optionality. To provide more context to fact-checks and claims, we are developing options for curators to leave comments for other users and edit fact-checks before they are sent out to the end user, so that we can improve the gaps in information that arise from this speed tradeoff.

We hope to test this with real users during a live political event in the near future, which will allow us to fine-tune our user interface and create an even faster, more intuitive user interface.

Acknowledgments: This project was done under the Bass Connections project “Data and Technology for Fact-Checking”, led by Jun Yang and Bill Adair.

DESIGN CHALLENGES
What do we optimize for when designing this web interface? The biggest challenge here was thinking about how we could design an interface that would be fast and intuitive. The two main challenges:

- Speed: since journalists are fact-checking in real time, we prioritized speed over all else, adding keyboard shortcuts and only offering the headlines of fact-checks to compensate, acknowledging that this came at the tradeoff of information clarity.
- Concurrency: since multiple fact-checkers could be performing the same fact-check at the same time, we need some way of indicating conflicting edits. UI features to solve this are listed to the right.

PROBLEMS WITH AUTOMATION
- Mistranslation: Google’s speech-to-text “mishears” a sentence. This is heightened by applause and politicians talking over each other
- Incorrect Grouping: Our speech-to-text component tries to identify where sentences begin and end, but often splits check worthy sentences into two seemingly irrelevant parts
- Difficulties with context: Our algorithm often fails to determine what “it”/“that” is referring to when a politician has mentioned a topic earlier in their speech

This chart shows the confidence of matches passed through our pipeline at 0.4 relevance

Mistranslation: Google’s speech-to-text “mishears” a sentence. This is heightened by applause and politicians talking over each other

Incorrect Grouping: Our speech-to-text component tries to identify where sentences begin and end, but often splits check worthy sentences into two seemingly irrelevant parts

Difficulties with context: Our algorithm often fails to determine what “it”/“that” is referring to when a politician has mentioned a topic earlier in their speech

Mismatching: Semantic similarities lead the algorithm to think a claim and fact-check are associated, despite them being completely different