Writing

The computer is intrinsically better at speaking than at listening. —Chris Crawford

Writing is the part of a conversational interface that people are exposed to the most. It's one of the most important parts but also one that's most at risk of being neglected. In fact when it comes to conversational interfaces, the writers are the ones who do a lot of the design.

Conversational interfaces are first and foremost composed of text. The text is split up into fragments because of both platform and medium limitations. Text fragments longer than a couple of hundred characters fall outside of this definition.

The flow of text can branch, often is conditional and parametric based on both local and global context. That makes writing for conversational interfaces complicated. It is in some ways even reminiscent of the literary technique called constrained writing.

Choose Your Own Adventure

Branching and dynamic textual experiences have been around for a long time in the form of interactive fiction and choose your own adventure books.
An ominous creaking of wood makes everyone fall silent in the treehouse. As you swing around and throw the switch on your flashlight, the smiling face of your classmate and best friend Joey is illuminated.

“Easy with that flashlight, dude,” Joey whispers. “I’m gonna need these eyes for later.”

“Not as much as you’re gonna need a cootie shot, Joey,” smirks Joey’s brother Mike. The seven occupants of the treehouse tremor with stifled laughter at Mike’s joke about Joey’s new girlfriend, Kali.

“Good one, Mike. But I think you only get cooties from kissing boys, the way you like to,” Joey hisses back.

Mike rocks forward from his Indian-style sitting position to stare at Joey, but a quick rap on the forehead with the heavy metal flashlight causes him to lose his balance and roll awkwardly onto his back like a chubby beetle.

“Sit down,” you say. “Everyone’s here now”

“Everyone but Brad,” says another friend, Scott, sadly.

“I know it’s everyone but Brad,” you sigh. Scott is notorious for making painfully obvious interjections such as this one. He is not the brightest friend you have. “That’s the reason

we’re here, Scott. We’ve got to find Brad.”

“How do we find him?” asks Michael.

Author’s note: This is Michael, not Mike. Yeah, you’ve got two friends named Michael. You can’t exclude people based on having more than one guy with the same name in your gang. That’s not cool.

“I don’t know. All we have are the notes we all saw, and the fact that his parents think he’s at summer camp. But we know better.” This last sentence comes out sounding a great deal more foreboding than you mean it to. “It’s time to move. Everyone get their bikes?”

Everyone nods. “And I brought some sodas for us to drink,” says your friend Mac.

“Sodas on our lake? The last thing we need is to crash and die covered in some sticky old soda,” says Mac’s brother Micah. “I brought Robinsons Fruit Shoot. It’s all-natural and it comes with a safety-tested kid proof sports top. So it’s good for us, and we won’t spill it everywhere like babies or girls.”

Turn to the next page.

As seen above each segment of story will have a number and at the end of it an instruction of which number to go to to follow a branch in the story. Interactive fiction works and the tools used by people in that field are useful references for work in conversational interfaces.

**Fallen London**

One recent example of the resurgence of these types of games is **Fallen London**.
Fallen London is a story game set in Victorian Gothic London. It has RPG-like elements, such as stats and attributes, that can be gained and lost. The main actions of the game revolve around storylets that offer choices and change your character. Based on your

<table>
<thead>
<tr>
<th>Actions</th>
<th>20/20</th>
</tr>
</thead>
<tbody>
<tr>
<td>BE EXCEPTIONAL!</td>
<td>Subscribe for a second candle and bring more stories every month.</td>
</tr>
<tr>
<td>FATE</td>
<td>Spend it...</td>
</tr>
<tr>
<td>WATCHFUL</td>
<td>81 +12</td>
</tr>
<tr>
<td>SHADOWY</td>
<td>97 +35</td>
</tr>
<tr>
<td>DANGEROUS</td>
<td>74 +12</td>
</tr>
<tr>
<td>PERSUASIVE</td>
<td>101 +17</td>
</tr>
<tr>
<td>SCANDAL</td>
<td>4</td>
</tr>
<tr>
<td>NIGHTMARES</td>
<td>3</td>
</tr>
<tr>
<td>DREADED</td>
<td>0 +1</td>
</tr>
</tbody>
</table>

**'Rattus Faber: Threat or Menace?'**

The respected journalist, Mr Huffam, is writing an article on London's rats. "They have become a matter of public concern," he says, licking his pencil. "What is your expert opinion?"

Record this in your Journal

| “The rat is a noble creature!” | Intrepid. Ingenious. Industrious. It won't be a popular opinion, but you have come to admire them. | Go 0 |
| “The only good rat is a dead rat!” | A simple solution to an intolerable problem. | Go 0 |
| “Rats are people, too.” | Some good, some bad. We should afford them the opportunity to prove themselves one or the other. | Go 0 |
| Unveil the ratty conspiracy! | The truth is worse than anyone suspects, and this is your chance to tell it! | Go 0 |
situation and your choices, different storylets become available, leading you through the story of Fallen London.

That multitude of elements makes Fallen London much more complicated than most chatbots and reminiscent mostly of the game environments we talked about in the chapter on history. It is useful to look at and draw inspiration from.

**Writers**

The demand for good copy in websites and applications has created opportunities for writers. Companies increasingly employ literary authors, comedians, and poets for this task. But just because somebody is a good writer doesn't mean they'll be able to add value to a digital production process.

In particular, writing for conversations requires somebody who — in addition to writing — can also deal with the complexity and the dynamic requirements. This person will need to create copy that has a coherent narrative but also makes sense for the overall system. They'll need to produce this in a format that others in the team can work with. They'll also need to be able to check whether their writing has made it into the product as intended. If it hasn't, their job will be to fix it.

Many projects will be tempted to offload the writing work on somebody else who's already a part of the team. This is inadvisable because both the type and the amount of work impose a large additional burden. It's not something that you can do (well) on the side.

Looking at the requirements above, it may seem like a daunting task to find people who are good at this skill, but it's by no means impossible. You'll be looking for writers who have proven experience in mixed media environments. This can be writing for games or for social media projects — anything where the writing is not static but instead a reaction to what the user does.

Rather than seeing it as a burden, getting a writer on board is a great opportunity to diversify your product team. The person you choose will most likely be somebody with a different skillset and background. In that way, they'll not just contribute their part but they'll also improve the project as a whole.

**Tone of Voice**
One of the first issues when writing for conversations is to figure out who is sending the texts. The system can send bare text, but a core part of a conversational system is the idea that there's somebody you're talking to. You'll need to figure out who or what that thing on the other side of the connection is.

You'll also need to figure out what tone of voice makes sense for your system to talk with and what that tone is backed by. This depends on how much you want to humanize the character of your system. Some chatbots pretend to be some kind of person and that influences how they respond.

If you don't already have a character design to work with, a basic sketch of who's talking may suffice. It should include some keywords and what kind of things this person will or won't say, in order to provide a rough picture. You can then use this picture to write out a segment of the interaction and try out how that feels.

**Dynamic Text**

There are two levels of dynamic text in a conversational interface: changes in a single message, and changes of which messages somebody receives.

The pieces of text in a single message can change. An example is of how your system would respond to a greeting. If it knows the user's name, it would reply: "Hello [first_name]!" This is a basic but realistic example. More situations like this — and especially more elaborate ones — will increase the complexity of the system.

These messages will depend either on what you know about the user (the user state) or where the system is at the moment (the system state). As a writer, you can't assume which state variables may be available at which moments. Additionally, some state variables may be missing in some situations.

This is why it's so important to work closely together with the rest of the team. This holds true both for the initial flows of conversations as well as the content of the messages later on. In a moving system, it'll be impossible to document this perfectly, so there will always be a need for working together. Being in direct communication with a team to create and check things together is the preferred approach.

Which message somebody receives at a given moment depends on the conversation flow and what follows what. I've discussed how to set up the general conversation in the chapter on prototyping. The writer should be involved here as well, as the sequence of
messages will inform the narrative as a whole. Any improvements here will have an effect that's felt throughout the system.

The writer needs to keep the narrative consistency in mind. This ranges from how things fit globally to how messages connect to each other. People going through the system using different paths need to have a good experience. This is something I'll talk about during the chapter on testing. A large part of building a conversational system is making sure the system says sensible things based on what came before.

**Rich Media**

Something that requires special consideration when thinking of a content strategy for conversations is that many platforms support some kind of rich media in their messages. Most basic of these are of course types of images that services may send to users. These could of course potentially be animated GIFs. What happens very seldom is that users can send images back to the system (like we did in Free Birds). A graphic element that is universally supported but very inconsistently rendered is putting emoji characters in the output text.

Richer media such as video and audio are rare but they are supported on some platforms. Depending how they are exactly supported, the way these media are played can also vary a lot.

For multi-channel application the lack of consistency of which media are supported in what way is an issue that requires consideration.

**Process**

The performance of any writer is determined in large part by how your development process is set up. A good process will allow the writer to grow and improve while they do the same with your product. A bad one will have them struggling to accomplish even the most basic tasks.

You'll want to brief them on the concept and the design notes of the conversational interface you'll be working on. The earlier in the process this person is on board, the larger their contribution can be. They should then work on the general direction of the story.
At some point, they'll start writing. You'll want them to do this in a way that the rest of the team knows how to work with. Doing the initial extra work to set up this process will be well worth it.

Along the way, you'll want to save your developer(s) from having to copy-paste a bunch of text into the platform. Ideally, the writer will be able to put their copy into its definitive place on their own. If that's not possible, they should at least work in an environment and format that's easy for the rest of the team to work with as well. This probably excludes tools such as Microsoft Word.

The writer will also need to be able to see their copy on the platform quickly. Writing in a text editor and receiving the same copy in chat are two very different things. Seeing it live is essential for being able to get the feel and length of the text just right. The ideal way to do this is by giving the writer a live environment they can put their copy into and test on. Otherwise, working with short release cycles isn't ideal, but it's doable.

Writers need to be able to change the things that need changing on their own. It doesn't matter how lightweight your issue tracker is. Communicating copy edits over any channel is a costly and time-consuming affair. It also carries with it many opportunities for misunderstandings.

Even if everyone on your team is able to work in the same environment, other stakeholders may want to check on progress and vet the writing. Many non-technical people have an intrinsic affinity for text and will naturally focus their attention on that part of the project. Your process should make it easy for them to go through all the copy and give feedback. Your overall planning should afford time for lots of copy changes during the project, especially at the end.

**Tools**

There are plenty of specialized writing tools I found useful. Many of these tools are free to use and try out — it's up to you to decide which of these, if any, work best for your project.

**Quip**

Quip is an online suite of office tools that's a lightweight alternative to the more established offerings. The Quip document editor lets people write a document together and discuss specific parts of it in comment threads. What sets it apart from, for instance,
Designing Conversational Interfaces

Google Docs, is its minimal formatting. This allows everybody to focus on the content.

Writing a conversation in a normal text editor requires a very rigid structure for it to fully specify the conversational flow. Otherwise, you should accept that what you write down is only a very general sketch of the final experience.

We used Quip a lot while developing Free Birds. It provided us with a quick way to work together on the copy. However, synchronization and duplication of work did take their toll in the long run.

The main story flow for Free Birds was linear, as is often the case for games. The production of content in games is costly. Every branch multiplies that cost by two going forward. Game design is the compromise of limiting that branching while maintaining the feeling of user agency.

Free Birds contains a linear back-and-forth between the player and game characters. Where there are multiple options, they generate different responses, but they don't change the course of the story. In Quip, for every option, we specify the button value and the message that it'll echo into the conversation.
This is similar to the approach used by the Walking Dead and other games. The choices players make don't branch the story, but they do change the responses (and in doing so, the feel). Changes in dialogue are much more affordable than new game segments, and this makes player choices feel consequential without branching the game's paths.

**Gingko**
Gingko is another tool we used in one of our own projects (KOKORO, a mental health coach). Gingko is a new tool that lets you shape your ideas and writing using a hybrid of lists, branches, and cards. Gingko has a learning curve, but it's a novel approach to organizing large and sprawling writing projects.

The editor's branching structure makes it somewhat suitable to author conversational interfaces in. The major issue is that you can only branch, which makes it impossible to loop back in a dialogue.

For our project, after a quick initial writing round, we translated the result into our own custom Javascript engine. After some iteration in the live environment, another editing pass in Gingko finalized the copy. Any changes after that were made exclusively in the code environment.

We solved the issue of not being able to loop by numbering the nodes in Gingko and then referring back to them. This worked, but it was a burden. As such, I wouldn't recommend doing this for larger projects. However, it's a good way to get an initial idea across and to continue building on it.

**Mindmup**

You can also sketch out the skeleton of a conversation using mind mapping tools such as Mindmup (though there are many, many others to choose from).
A mind map of a conversation won't look like a conversation and won't be experienced as such by most people. This only makes it suitable as a communications medium inside the product team, and poorly suited to communicating the goals and experience you're going for to external stakeholders.

**Twine**

Twine is an open-source tool for writing interactive non-linear stories. It provides a powerful and user-friendly tool to quickly prototype conversational experiences.

A chat is nothing but a specialized version of an interactive fiction experience. Interactive fiction consists of bits of story where options move you to different story locations. We can repurpose that by pretending the options are user responses, which take you to the system's next message.

Twine is customizable, though not easily so. The default presentation format doesn't look at all like a conversation, and it doesn't have message bubbles. However, there's no reason that it couldn't be made to look like that.

Given the suitability of Twine to build these kind of experiences, there are attempts to run a Twine story in a messaging client as well. ([Storygram](https://storygram.io) is one example.)
**Twee2**

*Twee2* is a text-based format to create Twine stories. This bypasses Twine’s main graphical user interface.

Twee2 is similar to *ink*, as it also allows you to create interactive fiction. The format is simpler and the setup of the system looks like it's more suitable for customization.

```plaintext
::StoryTitle
Escape from Earth

::Start
The spaceship is fuelled-up and [[ready to go->Ready]].

::Ready
You've still got a few seconds before you absolutely have to [[get on board->Board]], if you'
```

The syntax is very simple. Nodes are labeled with `::name`, and pieces of text can link to them using double brackets, like `[[name]]`.

Twee2 compiles into an HTML file that any web browser can display. Both the simplicity and the portability of the format make it appealing and suitable for prototyping a conversational interface.