CAREER (Comprehensive Automated Resume & Employment Experience Renderer) Language Specification

Harry Albert

May 8, 2024

Introduction

Creating a beautiful resume is often tedious and time consuming, but this annoying task is unfortunately extremely important. Making your resume look professional and easily readable can be the difference between a recruiter considering you for a job and throwing your application away. Formatting a resume well entails taking the time to organize your experience, skills, and interests in an aesthetically pleasing way using an often uncooperative word editing software. This means that making a resume can be both time consumign and painful, while still mandatory for career goals.

CAREER solves this issue by automatically formatting your information for you, taking as input some simple text and outputting a well designed resume. CAREER will always output a visually pleasing and readable product, without the fuss and frustration of actually formatting everything yourself. This also allows for the easy modification of one's resume without the need to go back and reformat everything. In summary, CAREER saves time by making resume creation a breeze, democratizing quality resume creation by putting a well formatted resume at anyone's fingertips.

Compiling a CAREER formatted text file will result in a well formatted LaTeX file, with the same name as the inputted text file and in the same location. If you have pdflatex installed (`https://tug.org/texlive/`), this LaTeX file will be automatically compiled to a pdf file (again, with the same name and in the same location as the inputted text file). You can also copy and paste the resulting LaTeX file into overleaf if you don't want to download a compiler.

Design Principles

The main goal of CAREER's design is to be easily readable and modifiable by the user. Any user who has skimmed over CAREER's documentation should be able to easily read and understand a resume written in CAREER. Users should be able to go back to their resume document after not looking at it for a year and quickly find and update whatever has changed in that time. This means that all of CAREER's commands and functions should be easily readable and understandable, ideally with little reference to the documentation. The other main design principle guiding CAREER is conciceness. One of the problems CAREER is solving is that resume writing is tedious and time consuming. Requiring the user to write lots of boiler plate code to create a CAREER resume would not fix this problem.

Examples

Example 1:

This is an example of a very simple resume that one could imagine a higschooler making. While this resume is very spare, it is still well formatted, and could serve as a good starting point to build off of in the future.
To run: Go into './code/lang' and run 'dotnet run ./examples/example_1/example_1.txt' (Note: when you copy and paste from latex, the underscores are replaced with spaces. Make sure to include the above underscores in your terminal command).

*HEADER Harry Albert

*BOLD Education
*ITEM Lenox Memorial High School, class of 2020
*ITEM Lenox Memorial Middle School

*BOLD Work Experience
*ITEM Life guard for Tanglewood
*ITEM Food service worker for Tanglewood
*ITEM Volunteered for Kid's Place in Pittsfield

Example 2:

This example program is a more complex iteration of the above problem. One might imagine this as the second version of a student's resume once they've entered college. This example uses a bit more of the formatting options available in CAREER and is thus more complex, but still very readable.
To run: Go into './code/lang' and run 'dotnet run ./examples/example_2/example_2.txt' (Note: when you copy and paste from latex, the underscores are replaced with spaces. Make sure to include the above underscores in your terminal command).

*HEADER Harry Albert
*SUBHEADER (111)111-1111
*SUBHEADER hsa1@williams.edu

*SECTION Languages & Technologies
*ITEM Javascript/Typescript, Python, HTML + CSS, F#, Java, C
*ITEM React, Scikit-Learn, PyTorch, SolidJS, REST APIs

Example 3:

This final example makes use of all of the functionality available in CAREER to make a complex and well formatted resume. However, this resume is still very readable and editable over time. One could imagine this as the resume of a graduating college student or working professional. This example is actually a shortened version of my Resume. If you want to see a full version of this resume written and compiled with CAREER, go to './code/lang/examples/resume'.
To run: Go into './code/lang' and run 'dotnet run ./examples/example_3/example_3.txt' (Note: when you copy and paste from latex, the underscores are replaced with spaces. Make sure to include the above underscores in your terminal command).

*HEADER Harry Albert
*SUBHEADER (413)770-6129, HarryAlbert364@gmail.com
*SUBHEADER *LINK[https://harryalbert.dev], *LINK[https://www.linkedin.com/in/harry-albert/]

*SECTION Education
*SUBSECTION_TITLE [Williams, MA] [Williams College] [Fall 2021 – May 2025]
*ITEM anticipated B.A. with double major in Computer Science & Philosophy
*ITEM Teaching Assistant for introductory and upper-level CS courses over five semesters

*SECTION Employment *SUBSECTION_TITLE [Software Engineer, Intern] [Bubble] [Summer 2023]
Bubble is a no-code web development platform. I worked on the Growth Team
*ITEM Ideated and executed project to refresh new hire onboarding, used by all new hires at Bubble, and created universal set of standards for writing and maintaining documentation
*ITEM Contributed to multiple successful experiments, including improving the editor UI and user journey

*SUBSECTION_TITLE [Software Engineer, Intern] [Meta] [Summer 2022]
Service Efficiency & Capacity Tooling Team
*ITEM Created two dynamic data visualization projects used by many engineers outside of the team
*ITEM Interviewed Meta engineers and gathered feedback in order to tailor designs to, and advocate for, their needs

*SECTION Projects
*ITEM *BOLD [*UNDERLINE ML Lung Cancer Prediction Tool]: Created a neural network to predict lung cancer based on several easily self-diagnosable conditions. Published a website where users can interact with this model to democratize access and allow users to play with the model. Available at *LINK[https://lung-cancer-risk-assessment-tool.vercel.app/].
*ITEM *BOLD [*UNDERLINE Booktrak]: Independently designed and created Booktrak for WSO (student run website for Williams College). Booktrak is a platform where Williams students can buy and sell used books, and link their book listings to specific classes in order to better facilitate used book sales. *LINK[View a demo video here][https://drive.google.com/file/d/1y5jxcYpGfvBr2p4P4GuF56aErHf5KXS2/view?usp=sharing].

Language Concepts

The core concept of this language is that there are modifiers and text. Text is a primitive, and it is defined as any string that is not a modifier. For example, in the line ”HEADER Harry Albert,” ”Harry Albert” is the text. Modifiers are (conventionally all caps) functions, indiciated by *, that change how text is displayed. For example, the SECTION modifier bolds the text that follows it and adds a horizontal line above it to indicate a new section. Ideally, users will be able to create their own modifiers to make the language more groweable. The ITEM function is a special modifier function that can only be applied to the start of a line. Finally, modifiers can be stacked. For example, ITEM creates a bullet pointed text, and BOLD (used within an item) creates a bolded piece of text. When brackets are included after a modifier, the modifier will apply only to the text included in the brackets. If brackets are not included, the modifier will apply to all text until a new line.

Formal Syntax

<expression> ::= <line>$^+$
<line> ::= <line_content> \n
<line_content> ::= *ITEM␣<formatted_text>$^+$
| *SUBSECTION_TITLE ␣<subsection_content>$^+$
| <formatted_text>$^+$
<subsection_content> ::= <string> | [<string>] | [<string>][<string>] | [<string>][<string>][<string>]
<formatted_text> ::= <modifier> | <string>
<modifier> ::= *<modifying_function>␣[<formatted_text>$^+$]
| *<modifying_function>␣<formatted_text>$^+$
| <link_modifier>
<link_modifier> ::= *LINK␣<formatted_text>$^+$
| *LINK␣[<formatted_text>$^+$]
| *LINK␣[<formatted_text>$^+$]␣[<formatted_text>$^+$]
<modifying_function> ::= HEADER
| SUBHEADER
| SECTION
| BOLD
| UNDERLINE

<string> ::= <str_char>$^+$
<str_char> ::= $c \in \{\text{all characters}\} \wedge c \notin \{\backslash n, *, [,]\}$

Semantics

Expression

Syntax: <line>$^+$
Abstract Syntax: Expression of Line list
Prec./Assoc.: 1/left
Meaning: An expression is simply a list of lines. Each CAREER ast is composed of one expression. This one expression holds all of the lines that were given as an input and will eventually be outputted as formatted LaTeX.

Line

Syntax: *ITEM␣<formatted_text>$^+$ \n
| *SUBSECTION_TITLE ␣<subsection_content>$^+$ \n
| <formatted_text>$^+$ \n
Abstract Syntax: Line of FormattedText list
Prec./Assoc.: 2/left
Meaning: One line in CAREER represents one input line in the given text file and one output line in the resulting tex/pdf file. A line is simply a list of formatted texts, meaning a list of modifiers and strings. A line is stopped by the new line character, and outputs a string with a new line character at the end.

Formatted Text

Syntax: <modifier> | <string>
Abstract Syntax: FormattedText of Modifier or String
Prec./Assoc.: 3/left
Meaning: FormattedText is essentially a container that can store either Modifiers or Strings. This is a necessary combining form because Modifiers must be able to be applied to other modifiers. Thus, modifiers are applied to lists of formatted texts, and each formatted text can store either a modifier or a string.

Item Modifier

Syntax: *LINE ␣<formatted_text>
Abstract Syntax: Modifier of string * (FormattedText list)
Prec./Assoc.: 4/left
Meaning: Item is a special modifier that puts an entire line into bullet point format. Because of this, Item (unlike other modifiers) cannot be applied anywhere within a line; it can only be applied at the start of a line. Thus, the line modifier has its own, special syntax that is taken into account by the Line combining form.

Subsection Title Modifier

Syntax: *SUBSECTION_TITLE ␣<string>
| *SUBSECTION_TITLE ␣[<string>]
| *SUBSECTION_TITLE ␣[<string>]␣[<string>]
| *SUBSECTION_TITLE ␣[<string>]␣[<string>]␣[<string>]
Abstract Syntax: Modifier of string * (FormattedText list)
Prec./Assoc.: 5/left
Meaning: The subsection title modifier is another special modifier. This modifier can also only be applied at the beginning of a line. It takes one to three arguments, all of which must be Strings. These arguments must be surrounded by brackets to deliniate between each argument. If there are no brackets, CAREER will assume the entire string after the modifying function is one argument. Given one argument, CAREER centers the given text, bolded and underlined, on a line. Given two arguments, the modifier places the first argument bolded on the left side of the line and the second argument bolded on the right side. Finally, given three arguments, CAREER places the first and third argument bolded on the left/right side of the line, and the second argument bolded and underlined in the middle.

Modifier

Syntax: *<modifying_function>␣[<formatted_text>$^+$] | <modifying_function>␣<formatted_text>$^+$ | <link_modifier>
Abstract Syntax: Modifier of String * (FormattedText list)
Prec./Assoc.: 6/left
Meaning: A modifier stores some modifying command and a list of formatted texts to apply this modifying command to. Each of these formatted texts must eventually evaluate to a string, and each of the strings in the formatted text list will be combined into one final string. This string will have the modifying function applied to it in the corresponding LaTeX code. For example, the modifying function BOLD applies \textbf to all of its inner formatted texts.

Link Modifier

Syntax: *LINK␣<formatted_text>$^+$
| *LINK␣[<formatted_text>$^+$]
| *LINK␣[<formatted_text>$^+$]␣[<formatted_text>$^+$]

Abstract Syntax: Modifier of String * (FormattedText list)
Prec./Assoc.: 7/left
Meaning: The Link modifier is a special modifier that can still be applied anywhere in a line. It is special because it can take either one or two arguments, and modifies its behavior depending on the number of arguments. If the user provides one argument, it is formatted as a link in the final LaTeX. If the user provides two arguments, the first argument is used as the text for the outputted link, and the second argument is used as the actual destination of the link (the first argument should be some display text, the second a url).

Modifying Function

Syntax: HEADER | SUBHEADER | SECTION | BOLD | UNDERLINE
Abstract Syntax: string
Prec./Assoc.: 8/left
Meaning: A modifying function is a command that tells CAREER how to format some following formatted text. What each of these modifying functions does should be clear from their naming. Modifying functions are preceded by the * character when used in a full modifier. This allows for the distinction between modifying functions and normal strings.

String

Syntax: <string>
Abstract Syntax: String of string
Prec./Assoc.: 9/left
Meaning: A string is, as you might expect, a string. It can contain any characters besides \n, *, [, and]. Each of these characters is used for a specific purpose in CAREER and thus cannot be used in basic strings.

Remaining Work

There are still some features that would be very valuable to add to CAREER. One such feature would be the ability to set important formatting values, like text size or line spacing, at the top of a CAREER file. This would allow for more customizable resume outputs. The user can do this one layer removed by editing the outputted LaTeX directly, but this removes some of the simplicity and ease of use that CAREER offers. Another valuable additional feature would be customizable modifying functions. Giving the user the ability to create functions that modify text using LaTeX code would, again, make CAREER much more customizable. It would also allow CAREER to grow more easily and naturally over time.
One feature that I did not implement from my prior draft is comments. The original idea was to allow users to add comments to their resume file in order to keep track of todos or other useful information. While this might have been helpful, it would require reserving another character or set of characters to denote a comment. I decided that reserving another character for this purpose was not worth the functionality that comments would provide. The main point of

comments in most languages is to help clarify hard-to-read code, but CAREER should be readable and simple enough that no comments are requird.