Creating Interactive and Visual Educational Resources for AI

Sameer Singh
University of Washington, Seattle WA
sameer@cs.washington.edu

Sebastian Riedel
University College London, London UK
s.riedel@cs.ucl.ac.uk

Abstract

Teaching artificial intelligence is effective if the experience is a visual and interactive one, with educational materials that utilize combinations of various content types such as text, math, and code into an integrated experience. Unfortunately, easy-to-use tools for creating such pedagogical resources are not available to the educators, resulting in most courses being taught using a disconnected set of static materials, which is not only ineffective for learning AI, but further, requires repeated and redundant effort for the instructor. In this paper, we introduce Moro, a software tool for easily creating and presenting AI-friendly teaching materials. Moro notebooks integrate content of different types (text, math, code, images), allow real-time interactions via modifiable and executable code blocks, and are viewable in browsers both as long-form pages and as presentations. Creating notebooks is easy and intuitive; the creation tool is also in-browser, is WYSIWYG for quick iterations of editing, and supports a variety of shortcuts and customizations for efficiency. We present three deployed case studies of Moro that widely differ from each other, demonstrating its utility in a variety of scenarios such as in-class teaching and conference tutorials.

1 Introduction

Perhaps more than any other field in computer science, teaching artificial intelligence benefits from a visual and interactive combination of mathematics, textual description, and programming code. From fundamental concepts such as Markov Decision Processes to the statistical and probabilistic descriptions in machine learning, mathematical equations underly much of AI. Similarly, programming is crucial to teach AI as well, ranging from boiler-plate code for evaluation and data processing, to basic search and optimization routines that underly much of artificial intelligence. It is further beneficial if the student is able to interact and visualize geometric spaces that are discrete (combinatorial state spaces for search, for example) and continuous surfaces (high-dimensional parameter vectors in machine learning, for example). For effective learning of concepts in artificial intelligence, students need access to interactive and visual learning materials that combine math, code, and text.

Even though the utility of visual and interactive resources for computer science educations in well known (Stein 1996; McIntyre and Wolff 1998; Naps and al. 2002), existing tools for creating educational resources, unfortunately, result in a disjoint collection of static teaching materials that the students process independent of each other. For instance, most of the current courses consist of notes and textbooks for purely static descriptions, presentation slides for a visual (and animated) overview, and code snippets for more interactive exploration of the ideas. These disconnected components, however, are not effective at communicating the ideas. It would be much more powerful if, for example, the code for generating figures in the textbook, or animations in the slides, was available for the students to explore or modify. Similarly, students would benefit greatly if code snippets are provided alongside the math and textual descriptions, in order to understand the code in context, but more importantly, enable them to change and see the effects. Although it is not impossible to create such resources in a format like HTML, the existing tools are not very friendly for AI instructors; on the other hand, individual type-specific tools (LaTeX for math/text, Powerpoint for presentations, and Programming-IDEs for code) are quite user-friendly. There's a need for resource-creation tools that can combine multiple media types, while retaining the ease-of-use and portability of existing editors.

In this work, we introduce Moro, a browser-based tool for creating visual and interactive educational materials for teaching artificial intelligence. Each Moro *notebook* consists of a number of cells, each of which can be formatted text, mathematical equations, programming code (that can be modified and executed by the student), or any embedded media (images, videos, PDFs, etc.). These notebooks can be presented to the students either as a web-page (akin to class notes or a textbook) or as animated slides (for in-class presentations), both of which retain the interactivity of the cells. The Moro editor has been created to be easy for instructors to use, including WYSIWYG previews, syntax highlighting, support for existing formats (Markdown, LaTeX, HTML), on-server code compilation and execution (allowing creation from any device), and broad customization and configuration options.

Moro is inspired by the interactive programming notebooks popularized by IPython (Pérez and Granger 2007), which also provide a mix of math, coding, and text, are used for authoring textbooks (Miller et al. 2011), and have found success in AI education (O'Hara, Blank, and Marshall 2015). Along with supporting these features, we designed Moro

directly for AI education: it provides a growing library of AI data structures for visualization, and supports full interactions on the client-side without requiring the students to install Moro.

Moro notebooks are being created for a number of different types of instruction materials; we explore a few case studies in the paper (§3). First, we demonstrate the use of Moro as conventional online documentation for the WOLFE probabilistic programming language that consists of introductory materials for probabilistic graphical models, optimization, and a number of demo applications of the language (spanning NLP, computer vision, and relational learning). Second, Moro is being used as the course material of graduate-level statistical natural language processing, presented as an interactive textbook for exploring various concepts such as NLP tasks (language models, parsing, etc.) and approaches (classification, structured prediction, learning, etc.). Finally, we also present the materials for the ACL[1] 2015 conference tutorial that consisted of substantial mathematical definitions, animated figures, and interactive code for topics such as matrix/tensor factorization and stochastic gradient descent. These case studies, together, demonstrate the flexibility of Moro as a tool for teaching a variety of artificial intelligence concepts, for very different audiences.

2 Visual and Interactive Moro Notebooks

The main unit of educational resources in Moro is a notebook. Each Moro notebook is similar to a single chapter in a textbook or a set of slides that cover a single topic. In this section, we present details on the different components of a notebook, and how they come together to create an interactive and visual teaching resource.

2.1 Notebooks

Each Moro notebook consists of a sequence of *cells*, where each cell can contain content only of a single type. We describe the different content types supported in Moro in the next section. This sequence of cells is used for laying out and formatting the content as the creator intends, for example a single slide in a Moro presentation often contains multiple cells. Although each cell is edited independently and can be moved around, they are not completely independent since, for instance, cells that contain code implicitly import the code from all previous code cells. Notebooks are not the highest level of organization in Moro, often multiple notebooks are used by creators, that are organized in a folder hierarchy that is presented within Moro in an intuitive manner.

2.2 Types of Cells

As described above, a cell is the smallest unit of content in Moro, and contains content of a single media type, similar to a paragraph or a set of equations. Along with a number of basic content types that we describe next, Moro also supports additional *layout* cells that define the document organization, for instance the section and sub-section dividers.

[1]Annual Meeting of the Association of Computational Linguistics, top NLP conference with more than 300 papers in 2015.

Text: Markdown Since text is clearly the most frequently used content type in almost all types of educational materials, we want creators to be able to easily write richly-formatted text without necessarily using a fairly complex language like LaTeX or HTML[2]. Instead, we use Markdown (Gruber 2004), an intuitive syntax that supports all kinds of basic text formatting (bold, italics, links, lists, etc.), along with features that AI instructors appreciate such as inline math using LaTeX, images, code blocks, and so on. Consequently, text cells in Moro are much more aesthetically pleasing than just unformatted text.

Math: LaTeX Since mathematical equations are fundamental to most topics in artificial intelligence, LaTeX (Lamport 1994) has been widely adopted by the field. The math cells in Moro also use the LaTeX syntax, allowing authors to write set of equations as easily as they would when authoring papers. Since Moro is browser-based, it uses an external library for rendering LaTeX in HTML (Cervone et al. 2009) that supports features such as equation numbering and labeled references from other cells.

Programming: Interactive Code Many of the crucial concepts in artificial intelligence can be effectively conveyed by programming, which is why most AI courses contain substantial programming components. Allowing students to read and modify code is one of the most important teaching tools available to the AI educators, however this practice has mostly been restricted to assignments. To take the next step of bringing interactive code to the classroom, or to incorporate it into the course materials, educational resources need to support two crucial components: (1) Visualization: the output of executing code blocks should be visually engaging, such as images, animations, plots, etc. instead of plain text *print* statements, and (2) Interaction: users should be able to modify code and regenerate the output as quickly as possible, for instance the instructor presenting modifications in-class, or student trying their own changes when learning.

This is the core functionality of Moro that differentiates it from existing content creation tools, and makes it suitable for AI educators. Moro notebooks can contain cells with program code[3]. The code in these cells is executed (on a remote server) at a click of a button, and the output is rendered immediately in the notebook itself. In order to encourage visually engaging output, the results of the code execution is HTML, thus allowing instructors to easily create visual and richly-formatted representations (Moro currently provides implementations for common data structures used in AI such as matrices as images, directed and undirected graphs, line/scatter/bar charts, search trees, graphical models, NLP annotations, and others). This combination of remote execution of code and HTML rendering of data structures is a powerful combination that enables instructors to create educational resources containing fully interactive programming content.

[2]Although Moro supports those as well.

[3]Moro currently supports Scala (Odersky and al. 2004), a functional language popular in machine learning and NLP, but extensions to other languages is planned. It is worth mentioning that IPython also supports Scala (Archambault 2015).

Other Types Although the content types so far cover a majority of what AI educators need, we don't want them to be constrained in any way when creating their materials. To this end, Moro supports almost limitless customizability and extensibility via cells that can inject arbitrary HTML and Javascript code into the notebooks. Further, we also provide an embedded viewer that supports many file types such as PDFs, videos, audio, Office documents, and so on. That said, we anticipate these features to be only used rarely.

2.3 Viewing Notebooks

Education materials are consumed and presented primarily in two different modes: (1) as reference reading material for individual students, and (2) as a slide deck presented to an audience by the instructor. We support these modes by allowing every Moro notebook to be viewed either as a long-form document, or as a presentation. Any overhead in installing viewers for education materials can be an obstacle to learning, and requires more effort on the instructors part, thus we want to make the notebooks as easy as possible for students to view. We take a browser-centric, lightweight-client approach, allowing Moro notebooks to be viewed (and as we will see later, created) on any device capable of running a modern browser, including phones and tablets, both as long-form documents and as presentations. Another advantage of having browser-based layout is that Moro notebooks can be further customized by the instructor, if desired, using CSS.

Long-form Document View In this view of a notebook, all the cells of the notebook are laid out sequentially, one after the other. The resulting output is similar to textbooks, class notes, or online reference web-pages. However, the code cells are still interactive, allowing modification and execution by each viewer of the notebook independently, i.e. multiple students can make their own edits and executions to the same notebook. Further, in the rare case offline access required, this view is also printer-friendly and supports export to static document formats like PDF.

Presentation View When teaching a large audience, such as in a classroom, notebooks can also be viewed as a presentation. Moro uses a browser-based presentation library (Hattab 2011) in order to support this, which provides features as slide animations, transition animations, keyboard shortcuts, overview, themes, touch control, speaker view, and so on. Further, even the presentation view supports interactive programming, allowing the presenter to modify code and regenerate output during the presentation.

3 Examples of Moro Use Cases

In this section, we demonstrate some of the variety in scenarios of teaching in artificial intelligence for which Moro has been used to create educational materials. The case studies described here vary considerably, ranging from conventional course material, to online documentation, and to a conference-style tutorial. Through these case studies, we will explore the utility of Moro in allowing instructors to cover a variety of topics in artificial intelligence, explore different visual and interactive elements in their presentations, and flexibility and customizability of Moro notebooks to different mediums.

3.1 Case I: WOLFE Documentation and Demos

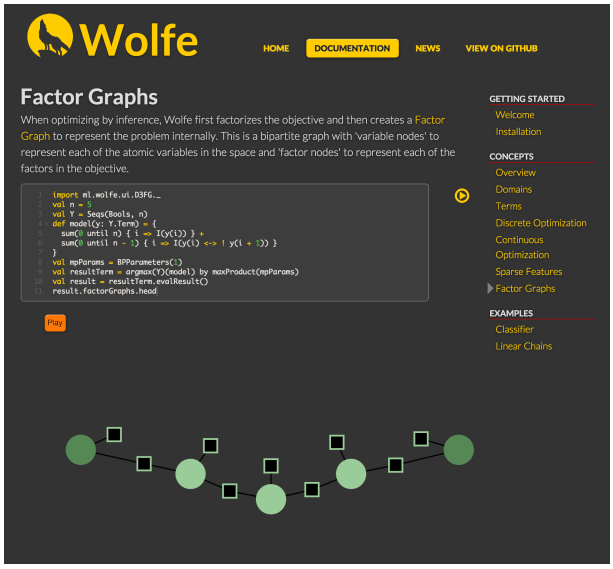
Creators: *Sebastian Riedel, Tim Rocktäschel, Sameer Singh, Luke Hewitt, Jason Naradowsky*
Link: `http://www.wolfe.ml`
WOLFE is a probabilistic programming language that can be used to declaratively define machine learning models and optimization approaches, thus allowing deployment of sophisticated algorithms even by people unfamiliar with artificial intelligence (Riedel et al. 2014). Moro has been used to create tutorial documentation for the language, which includes not only the syntax of the language, but also explanation of the underlying machine learning concepts (see Figure 1 for screenshots). Even though online documentation does not usually fall into the standard view of teaching, the interactive nature of the code, and the combination of math, text, and rendered output, made these Moro notebooks perfect tools for teaching both the language and machine learning. The feedback from the users has been quite positive, especially the interactive exploration of factor graphs (as shown in Figure 1a) provided useful insights into conditional independencies in complex probability distributions.

3.2 Case II: Statistical NLP Course Material

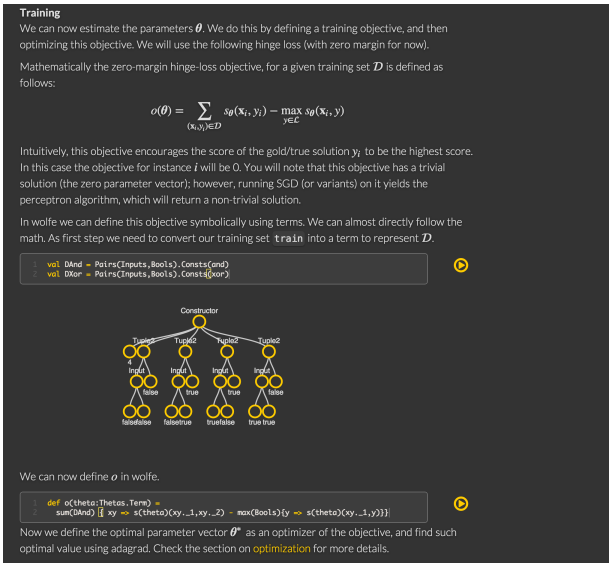
Creators: *Sebastian Riedel, Matko Bošnjak, Tim Rocktäschel*
Link: `github.com/uclmr/stat-nlp-book`
The second use case we present here is a graduate-level course on statistical natural language processing that covers NLP tasks (such as tagging and dependency parsing), machine learning methods (such as linear chain CRFs and matrix factorization), and optimization (such as dynamic programming and gradient descent). Moro is used for creating an online textbook for the course, with a notebook for each chapter (see the screenshot in Figure 2). Not only is Moro able to recreate a similar formatting as a book, but further, the interactivity provided by the code cells allows the students to not only observe the code behind all the illustrative examples, but also modify the example and regenerate the figures right in the textbook.

3.3 Case III: ACL 2015 Tutorial

Creators: *Guillaume Bouchard, Jason Naradowsky, Sebastian Riedel, Tim Rocktäschel, and Andreas Vlachos*
Link: `github.com/uclmr/acl2015tutorial`
As a final example, we consider the use case of presentation to a large audience, specifically the ACL 2015 tutorial on "Matrix and Tensor Factorization Methods for Natural Language Processing" that was presented on July 26, 2015 in Beijing, China to the conference attendees. The presentation view of Moro was used to created the slides for the entirety of the 3.5 hour long tutorial, which contained a combination of equations, text, figures, and code as shown in Figure 3. Mixing these different content types in the same editor was found to be useful by the tutorial presenters, and the ability to modify code during the presentation allowed the presenters to investigate novel teaching strategies that would not have been possible when using conventional presentation tools. The attendees felt that they benefited from the interactivity, and appreciated the ability to download and replay the slides on their own after the tutorial.



(a) Interactive page with a combination of code and the visualization of the factor graph generated by the code. The students can modify the mathematical definition of the model, and explore how it translates into conditional independences.



(b) Another section of the documentation showing a mixture of math and text, with a tree representation of data structures. The students can not only regenerate the tree by modifying the code, but the visualization itself supports interactions such as panning and zooming to navigate more complex trees.

Figure 1: WOLFE Machine Learning Toolkit Documentation

4 Creating Notebooks

So far, we have highlighted the various features of Moro notebooks that make it useful for teaching AI, and studied a few examples of how Moro is being used today. However, often adoption of any format depends crucially on its ease of use, more so in teaching since the instructors differ considerably in their backgrounds and expertises. With this in mind, Moro notebook editor has been created to be extremely easy to use: it uses WYSIWYG[4] UI principles to give an accurate representation of the notebook, uses syntax highlighting and keyboard shortcuts customized to each content type, and supports browser-based interface and a portable file format for easy creation on any device. A screenshot of the Moro notebook editor, shown in Figure 4, highlights many of these features, which we will also elaborate upon next.

4.1 WYSIWYG Interface

The editor interface is laid out with the cell editors on the left (text fields for entering the content), and the rendered version of the cell displayed on the right. As the author changes the content of the cell, for example entering Markdown in the first cell in Figure 4, Moro renders the output immediately on the right. This constant rendering allows instructors to detect errors and iterate quickly on the content. Further, the conversion of the cell code to HTML is carried out remotely on the server, thus requiring minimal computational requirements from the client's browser.

[4] "What You See Is What You Get"

4.2 Cell Editors

Since the cell editor (left-side) is the where the content creators are likely to spend most of their time, we've introduced a number of features for further ease-of-use. The editors support a wide variety of regular text-editing keyboard shortcuts, along with extra ones specific to Moro such as navigating between previous/next cells, regenerating the output of the current cell, changing the content type of the cell, splitting the cell, and so on. Each cell editor also adapts to the content type, using the appropriate syntax highlighting and autocompletion, making them quite similar to popular editors of the specific content type.

4.3 File Format and Portability

The underlying format of Moro notebooks is JSON (ECMA 2013), a human readable (and editable) format for structured data. Using this text-based format not only integrates with revision control systems and external text-friendly editors, but also opens up the possibilities of interfacing with other systems using existing JSON libraries. Further, since only the cell content is stored, and not the actual rendered HTML, the Moro notebooks are quite small in size, and thus portable.

4.4 Installation

A difficult and complicated installation of tools causes an unnecessary overhead for the student's learning, and requires additional effort on the instructor's part to provide support. We have attempted to make Moro quite easy to install; on any Linux/Unix shell, including OSX and Cygwin on Windows, Moro can be installed and run as a web-server using $2 - 3$

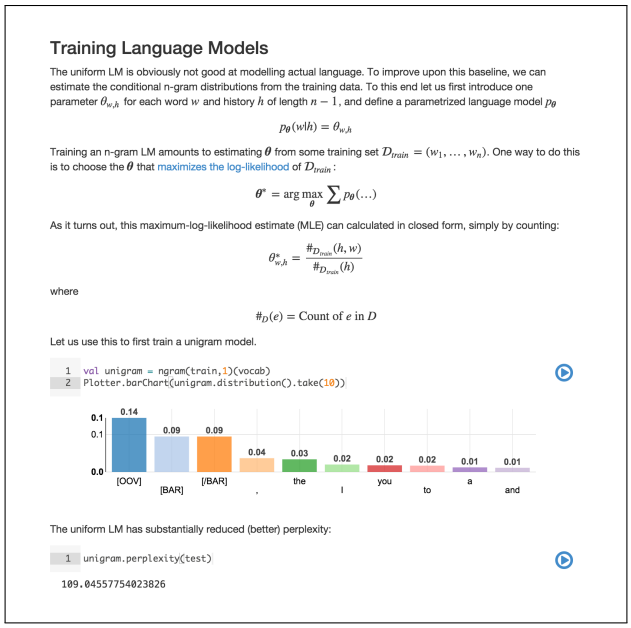


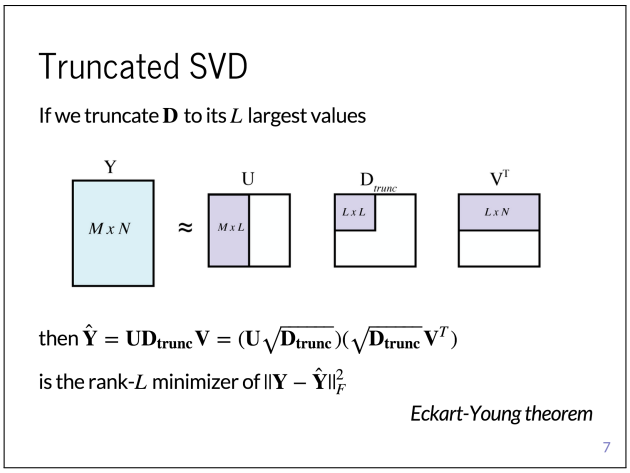
Figure 2: A section of the statistical NLP graduate course material showing math, text, and code, where the bar chart has been generated directly from the code, and thus can be modified. Data used to generate the distribution appears earlier in the notebook.

lines of code. Note that the installation is required primarily for creating, editing, and displaying notebooks; in many instances only the instructor needs to set up this server, and the students are able to view notebooks on any browser.

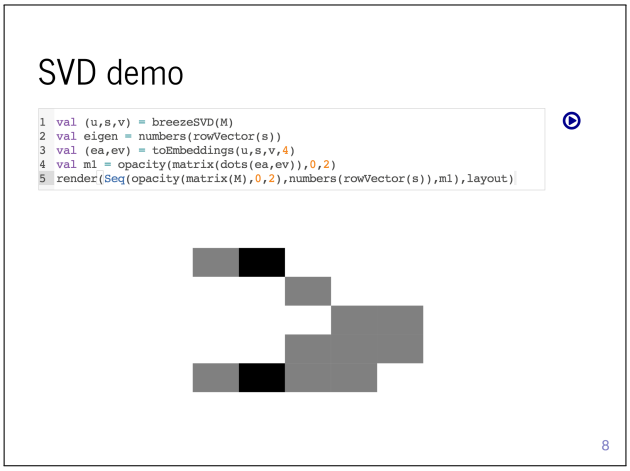
5 Conclusions and Future Work

We are motivated by the need for interactive and visual educational materials for teaching AI that are effective pedagogical resources (Stein 1996; McIntyre and Wolff 1998; Naps and al. 2002), however are extremely difficult to create using existing tools. To address this concern, we propose Moro, an in-browser creation and presentation tool that allows instructors to easily create such visual and interactive pages. In particular, Moro notebooks allow instructors to combine together math, text, and images with interactive code blocks that visualize AI data structures. We highlighted three use cases of Moro that cover different modes of AI education: course material, online documentation, and tutorial presentation. Moro source code, along with the case studies, is open source at `http://wolfe-pack.github.io/moro` under the BSD license.

There are a number of future avenues we plan to explore. To increase the coverage of instructors with different programming backgrounds, we would like to expand the programming languages that Moro supports. Given Moro's REST-ful API, we do not anticipate this to be a significant challenge. We are also going to investigate data structure visualizations commonly used in teaching artificial intelligence beyond the ones currently supported by Moro (matrices, graphs, trees, plate models, and plots), including popular



(a) Slide with combination of text, figures, and equations



(b) Slide with interactive code block, modifying which during the presentation generates different output matrices.

Figure 3: Slides from the ACL 2015 Tutorial

application-specific structures such as the grid-world and computer vision (Moro currently supports NLP tasks). Finally, we are actively seeking feedback from Moro users (students and instructors), and plan to improve Moro continuously to incorporate their suggestions.

Acknowledgements

The authors would like to thank Tim Rocktäschel, Luke Hewitt, Jason Naradowsky, Guillaume Bouchard, Matko Bošnjak, Andreas Vlachos, and Vivek Srikumar for ideas, testing, and bug reports for early versions of Moro. This work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, in part by the Paul Allen Foundation through an Allen Distinguished Investigator grant and in part by a Marie Curie Career Integration Fellowship.

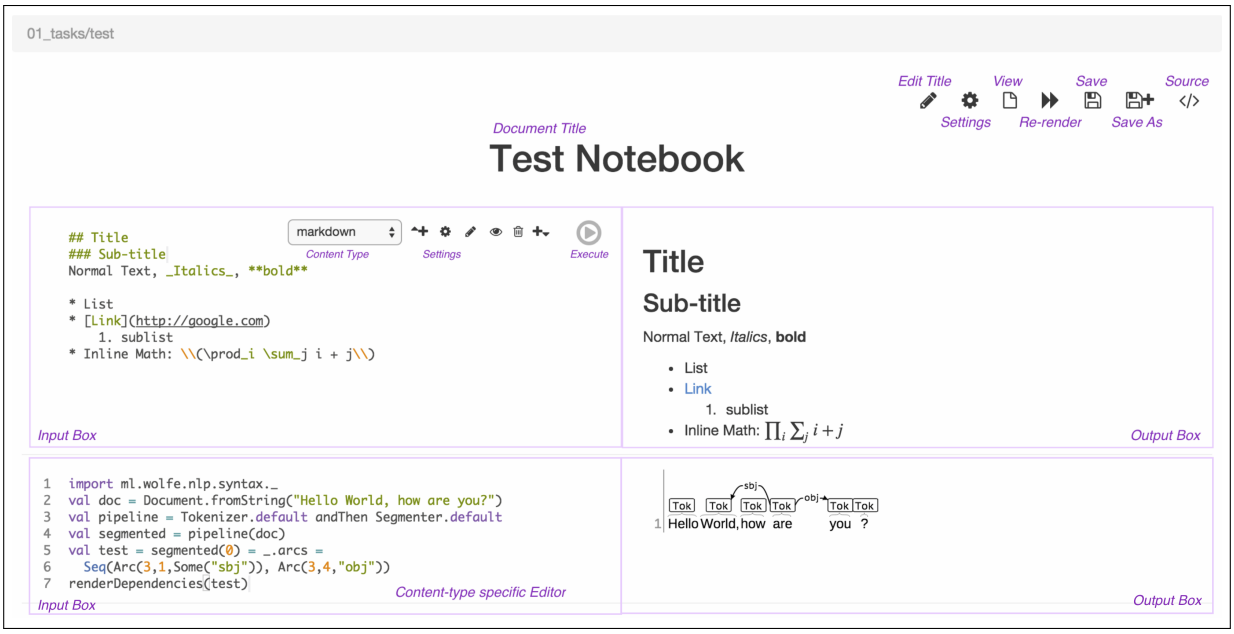


Figure 4: **Moro Notebook Editor**: Screenshot of the Moro notebook creation interface, with some of the features such as WYSIWYG interface, cell mode selection, cell and notebook configuration, etc. highlighted in purple.

References

Archambault, A. 2015. Jupyter Scala. `https://github.com/alexarchambault/jupyter-scala`. [Online; accessed 30-Nov-2015].

Cervone, D.; Sorge, V.; Perfect, C.; and Krautzberger, P. 2009. MathJax. `https://www.mathjax.org/`. [Online; accessed 15-Sep-2015].

ECMA. 2013. The JSON Data Interchange Format. Technical Report 404, ECMA Standards.

Gruber, J. 2004. Daring Fireball - Markdown. `http://daringfireball.net/projects/markdown/syntax`. [Online; accessed 15-Sep-2015].

Hattab, H. E. 2011. reveal.js - The HTML Presentation Framework. `http://lab.hakim.se/reveal-js/`. [Online; accessed 15-Sep-2015].

Lamport, L. 1994. *LaTeX: a Document Preparation System*. Reading, Ma.: Addison-Wesley Publishing Company, 2 edition. Illustrations by Duane Bibby.

McIntyre, D. R., and Wolff, F. G. 1998. An experiment with WWW interactive learning in university education. *Computers & Education* 31(3):255 – 264.

Miller, B.; Ranum, D.; Guzdial, M.; Ericson, B.; and Thakur, V. 2011. Runestone Interactive. `http://www.interactivepython.org/runestone`. [Online; accessed 30-Nov-2015].

Naps, T. L., and al. 2002. Exploring the role of visualization and engagement in computer science education. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '02, 131–152. New York, NY, USA: ACM.

Odersky, M., and al. 2004. An Overview of the Scala Programming Language. Technical Report IC/2004/64, EPFL, Lausanne, Switzerland.

O'Hara, K.; Blank, D.; and Marshall, J. 2015. Computational notebooks for AI education. In *Florida Artificial Intelligence Research Society Conference*.

Pérez, F., and Granger, B. E. 2007. IPython: A system for interactive scientific computing. *Computing in Science and Engineering* 9(3).

Riedel, S.; Singh, S.; Srikumar, V.; Rocktaschel, T.; Visengeriyeva, L.; and Noessner, J. 2014. Wolfe: Strength reduction and approximate programming for probabilistic programming. In *International Workshop on Statistical Relational AI (StarAI)*.

Stein, L. A. 1996. Interactive programming: Revolutionizing introductory computer science. *ACM Comput. Surv.* 28(4es).