Tips about writing systems papers

by Lin Zhong, May 2015

If I can give one tip, that would be: *respect your readers*.

Put yourself into your readers' shoes

It is important to visualize your readers and how they read your paper. You hope they would enjoy your paper like you enjoyed the Harry Potter books; you hope they would study your work like you studied your Physics 101 textbook; at least you hope they would read your paper like how you read a classic paper recently. Such hopes are completely false, because nobody reads technical papers for recreation, you are obviously not writing a textbook, and more importantly, your paper is not yet published, let alone being a classic.

So *how does a reader read your paper?* Or more precisely, *how does a reviewer read your paper?* **They try to finish the job as fast as possible.** Peer-review is a service that does not pay. Reviewers try to understand the work as fast as possible so that they can do their job: write a review and decide if they want to champion for reject, accept or they don't really care. Often they have a dozen or so papers to review and likely are already behind the deadline. So you can imagine their most likely mood at this point: **impatient**, perhaps **irritable**. Well if you are really really lucky, one of your reviewers is reading your paper one month before the deadline on the beach with a glass of sparkling wine. But I can tell you that rarely happens.

Importantly, top publication venues usually have a very low acceptance rate, accepting one out of five submissions or less. That means the majority of the papers read by a reviewer are likely to be rejected. Therefore, it is not unusual a reviewer starts reading each paper with a subconscious bias looking for reasons to reject it. So much so that many TPC chairs these days try very hard to guide the reviewers to read submissions with a positive mindset and to focus on identifying values, instead of flaws. Until they are successful, you can safely assume your reviewers are **hostile** in that they are looking for reasons to reject your paper. Your job is to win them back.

In the rest of this article, *readers* include *reviewers*, the very first outside readers of your paper.

General techniques

Once you decide to make readers' life easier, you would immediately realize there are numerous ways, each of which, of course, requires you to spend more time with your draft. Below I summarize a few useful general techniques, roughly categorized as presentation and logic.

Presentation

- *Be concise*. Every word slows the reader a little. See if you can say the same thing with fewer words. Remove a word, sentence, paragraph, figure, section unless they are absolutely necessary. The same principle applies to conceptual units: remove a definition, system component, design detail unless they are absolutely necessary. Resist the urge to tell readers everything. Instead, tell them enough for them to appreciate your points.
- *Write top-down*. Always put your

point upfront because readers are impatient and skimming. This rule applies to structures at all levels from paragraph to section. A point is what you want the readers to remember, not a summary of the content but the conclusion supported by it. For instance, the point of the evaluation section is ``Experimental evidence demonstrates our solution advances the state of the art'', not ``We report extensive experimental evaluation that compares our solution against the state of the art.''

- *Use active voice*. You don't want readers to guess who did something or contributed an opinion. You don't want them to guess which component of your system is behind an action.
- *Use structure*. Your readers, Homo sapiens, have 5 to 9 entries in their short-term memory. Structure helps them abstract/chunk materials in order to use these entries efficiently. It also makes relationships between structural components obviously. A very long paragraph, a page of paragraph after paragraph, and a section of a large number (7 or more) of subsections are signs that structure may be necessary. The lack of structure often results from the lack of focus or abstraction. Focus requires one to ignore unimportant things; abstraction requires one to identify unifying themes, crosscutting principles, and high-level relations.
- *Use simple English*. Many graduate students incorrectly believe a good paper should read like the essays they studied for GRE. Remember the GRE essays were selected to evaluate your reading capability. You absolutely don't want your own readers to feel that you are evaluating their reading capability. Avoid long sentences; avoid complicated sentence structures; avoid arcane terms/idioms; avoid big,

showy words. You are welcome to try them after you have got some grey hairs.
- *Use graphs effectively*. Most readers pay attention to graphs. Make sure each graph deliver an important point. Make sure it can do so by itself (so make good use of its caption). Do refer to the graph in the text so that it assists understanding.

Logic

- *Don't surprise*. Set up the right expectation. Don't promise early on, e.g., in abstract and introduction, more than you deliver later, e.g., design, implementation and evaluation. A very common reviewer comment is ``I started reading this paper with excitement but the more I read the more I was disappointed."
- *Don't handwave*. Quantify a claim. For example, when you say ``our solution significantly outperforms the state of the art", don't stop there but quantify ``significantly" with ``improving the data rate by three times". For another example, when you say ``our design requires very small changes to the Linux kernel", quantify it with ``with only 3 added code line at a single location". Cite an authoritative source for a fact that is not yet widely known but key to your point.
- *Don't haunt readers*. As people read, they will have questions. Some questions may trouble them if not answered soon. I call such questions the *haunting questions*. You must anticipate such questions and answer them before a reader starts to wonder. If you can't answer them immediately, do acknowledge their existence and assure the reader you will answer them later with a forward reference.
- *Be consistent*. Consistency in naming

and style makes your writing easier to understand. Call the same thing the same name. If one figure shows the software stack vertically and another horizontally, you make readers' job more difficult. When figures of similar plots are grouped together, e.g., speed vs. number of processors for several benchmarks, make their axes consistent in terms of unit and range. When it is impossible, e.g., one benchmark has higher speed by orders of magnitude than others, highlight that inconsistency, e.g., pointing it out in the caption or using a different font size for the inconsistent axis.

Specialized techniques

Now I describe tips for important structural components of system papers.

Title

A good title captures both the problem addressed by the paper and the novel solution contributed. To avoid a title too long, it often suffices to capture the problem only.

Abstract and Introduction

Many readers only read the abstract and introduction. Many a reviewer makes up their mind after reading them. The introduction is so important that even the most hands-off advisors will read, revise, and sometimes rewrite it. I suspect that's a major source of surprises and inconsistencies. The introduction should summarize the entire paper crisply and bring out major points you want readers to walk away with.

- *Motivation*. There is a delicate balance one must achieve here with intended readers in mind. A problem well-known to your intended readers needs no motivation but a simple statement of its importance, e.g., ``Energy efficiency is a critical design concern for battery-powered mobile devices". An emerging problem anticipated by you or a small part of the community requires more explanation. However, the motivation should never be more than a small paragraph, simply because readers are impatient: they want to get to your problem statement quick. If you have a rather long motivation, either it contains a lot of obvious, well-known materials or you are solving a trivial problem.

- *Problem statement*. Don't wait too long to state the problem. If the motivation is a small paragraph, the problem statement can be the first sentence of the second paragraph. The problem statement must be obvious, prominent, explicit. For example, ``This paper answers the following question:...", "Our goal is to support concurrent computer vision applications efficiently", and "This paper presents our design and implementation of a system that supports concurrent computer vision applications efficiently." A common mistake is to place the problem statement at the end of a paragraph that elaborates the problem; instead, the statement should be placed at the opening of the paragraph. Remember to write *top-down*.
- *Prior work*. After the problem statement, you need to concisely explain why prior work is inadequate. There is no need to discuss related papers one by one here.
- *Key ideas*. Without reading the rest of the paper, a reader must appreciate why you are able to solve the problem while prior work falls short. Therefore, you need to explain your

key ideas, secret sauce, explicitly and prominently in the introduction. For example, "Our design is based on two novel insights. First,...", "We solve the problem with a clever synthesis of technique A and technique B."

- *Evaluation results*. Summarize how you evaluate your work and the main results. Be quantitative. Be upfront about negative results.
- *Contribution summary*. In order to help the reader, especially the reviewer, provide an itemized summary of your contributions. There should be no more than three or four items.

Background

Unless you are writing for a very narrow community, it is very likely your readers do not have all the background to appreciate your work. So educate them. A nicely written background section is always enjoyable. Indeed, learning something new so quickly is one of the perks of doing reviews. The challenge is: the background section must be self-contained, concise, clear. This often requires a deep understanding of who your intended readers are. For a paper submission, it often helps to take a look at the program committee and imagine which members are likely to review your paper.

Design

For a systems paper, it is often nice to describe design and implementation separately. The *design* section often presents ideas that are agnostic to any specific implementation. These include the overall architecture, principles and algorithms.

A common mistake of the design section is lack of structure, e.g., a section of seven or more subsections each describing a component of the design. As mentioned above, the lack of structure often results from the lack of focus or abstraction. For example of *lack of focus*, the authors simply describe each system components, regardless of how interesting/novel they are. If the system has a lot of components, this strategy leads to a flat section with many subsections. One naturally solution is to use a subsection to provide an overview of the design and then use subsections to explain only important components that are novel and interesting. Presenting the design by component is often a sign of lack of abstraction. System component is just one possible dimension along which you can present the design. There are other dimensions, e.g., design objectives and principles/invariants followed. Discovering these dimensions require you to think about your design in an abstract way.

Another common mistake is failure to place a design choice in the context, e.g., simply describing it. There are two types of context that add depth. First, how is your choice related to prior work? Perhaps no one has solved the exactly same problem but it is highly likely others have faced similar challenges in solving a related one. It is important to be aware of how others made the choice and explain how their choice affected your choice. This is also a perfect place to give prior work credit. Second, what alternatives have you considered? It is highly unlikely your choice is straightforward or obvious (if it is, you should not spend a lot of space highlighting it). Share with readers what other choices you have considered and why you passed them in favor of the one ended up in the design.

Implementation

The section of implementation should not be a boring description. It should present interesting things about your implementation. If there is nothing interesting, the section should be very brief, providing the necessary information for readers to assess the quality of your evaluation, which is presumably based on the implementation.

What are the interesting things about the implementation? As the implementers, you should know. For example, any nontrivial challenges have you had to overcome? How does the implementation have to deviate from the design due to practical limitations? Any platform-specific optimization?

It is important to let the readers know how significant the implementation effort is. You can quantify it with the number of lines of code and the number of man-hours.

Evaluation

Since this article is about writing, instead of research itself, I will focus on writing, assuming you have done all the right things in evaluating your work.

- *Retain a neutral tone toward your solution*. It is important to remember that the objective of evaluation is NOT to show that your solution works. Instead, it is to show how well it works, importantly when it works and when it does not.
- *Be explicit upfront about what questions you want to answer*. These questions are often related to the contributions you claim. For example, if you claim a novel optimization A that saves a lot of energy, the question to answer via evaluation would be "how much energy does A save? How does the saving depends on operational parameters?"
- *Acknowledge the limitations of your evaluation*. And explain how it may affect the conclusions you draw. When you acknowledge them, readers find you honest, trustworthy; when you

don't, they are more likely to consider the evaluation as contrived.

- *Share negative results*. A very common mistake is to only show positive results. Unless your solution always works, it is your responsibility to tell readers when it works and when it does not. In other words, you have to specify the *scope* under which your solution works. Sharing negative results not only adds depth to your paper but also impresses readers as being honest: too many times I have heard reviewers complaining the evaluation setup appears to be contrived, the benchmarks appear to be cherry-picked. To the contrary of the belief by many young researchers, sharing negative results actually strengthens your paper and earns respect from readers.
- *Provide necessary information for reproducibility*. The key of scientific and engineering research is reproducibility. That is, others should be able to follow your writing and reproduce your results. For software, it is nice to make it openly available, open-source or not. When machine configurations matter, you should consider providing a virtual machine with the same configuration as used in the reported evaluation. Where the physical setup matters, e.g., wireless channels, be specific about the physical parameters, e.g., providing a floor map with node locations precisely marked.

Related work

The related work section is important in many ways. It sets your work in the larger intellectual context and explains how it advances the state of the art. For novice readers, it points out a quality collection of good works that they can follow up. For expert readers, it is the perfect place to check the authors' mastery of the topic area. For the authors of the cited work, it is where their own work is discussed, credited and criticized. A well-written section of related work always wins respect from its readers. Below are a few tips.

- *Use structure*. Don't discuss related works simply one by one. Discuss them in groups. For each group, start with the common things of the group and then important ones one by one. Of course this requires you to understand how related works are related to each other and to your own work.
- *Be generous*. First, when discussing a related work, don't just focus on how it is different: start with how it is similar. Don't just criticize its shortcomings: start with its strength and value. Second, give benefit of doubt to a related work. Don't interpret it in favor of your work. For example, when a related work did not mention whether its idea works for iOS, you should not interpret that as the idea will not work for iOS. A general way to examine if your discussion is generous is to imagine if you were the author of the discussed related work, would you like the way it is discussed?
- *Be insightful*. Don't be content with superficial similarity and dissimilarity. Only when you point out something non-obvious, you earn readers' respect for your insight.

Recommended books

I ask all my students to buy and read the following two books:

- *The elements of style* by William Strunk Jr. and E. B. White (any edition).

- *Style: lessons in clarity and grace* by Joseph Willims and Gregory G. Colomb (any edition, I have 10th).

The following book is more specialized for Computer Science but I have not read it as carefully as the above two:

- *Writing for computer science* by Justin Zobel.
