How to Write an Abstract

Philip Koopman, Carnegie Mellon University
October, 1997

Abstract

Because on-line search databases typically contain only abstracts, it is vital to write a complete but concise description of your work to entice potential readers into obtaining a copy of the full paper. This article describes how to write a good computer architecture abstract for both conference and journal papers. Writers should follow a checklist consisting of: motivation, problem statement, approach, results, and conclusions. Following this checklist should increase the chance of people taking the time to obtain and read your complete paper.

Introduction

Now that the use of on-line publication databases is prevalent, writing a really good abstract has become even more important than it was a decade ago. Abstracts have always served the function of "selling" your work. But now, instead of merely convincing the reader to keep reading the rest of the attached paper, an abstract must convince the reader to leave the comfort of an office and go hunt down a copy of the article from a library (or worse, obtain one after a long wait through inter-library loan). In a business context, an "executive summary" is often the *only* piece of a report read by the people who matter; and it should be similar in content if not tone to a journal paper abstract.

Checklist: Parts of an Abstract

Despite the fact that an abstract is quite brief, it must do almost as much work as the multi-page paper that follows it. In a computer architecture paper, this means that it should in most cases include the following sections. Each section is typically a single sentence, although there is room for creativity. In particular, the parts may be merged or spread among a set of sentences. Use the following as a checklist for your next abstract:

- **Motivation:**
  Why do we care about the problem and the results? If the problem isn't obviously "interesting" it might be better to put motivation first; but if your work is incremental progress on a problem that is widely recognized as important, then it is probably better to put the problem statement first to indicate which piece of the larger problem you are breaking off to work on. This section should include the importance of your work, the difficulty of the area, and the impact it might have if successful.
- **Problem statement:**
  What *problem* are you trying to solve? What is the *scope* of your work (a generalized approach, or for a specific situation)? Be careful not to use too much jargon. In some cases it is appropriate to put the problem statement before the motivation, but usually this only works if most readers already understand why the problem is important.
- **Approach:**
  How did you go about solving or making progress on the problem? Did you use simulation, analytic models, prototype construction, or analysis of field data for an actual product? What was the *extent* of

your work (did you look at one application program or a hundred programs in twenty different programming languages?) What important *variables* did you control, ignore, or measure?

- **Results:**
  What's the answer? Specifically, most good computer architecture papers conclude that something is so many percent faster, cheaper, smaller, or otherwise better than something else. Put the result there, in numbers. Avoid vague, hand-waving results such as "very", "small", or "significant." If you must be vague, you are only given license to do so when you can talk about orders-of-magnitude improvement. There is a tension here in that you should not provide numbers that can be easily misinterpreted, but on the other hand you don't have room for all the caveats.
- **Conclusions:**
  What are the implications of your answer? Is it going to change the world (unlikely), be a significant "win", be a nice hack, or simply serve as a road sign indicating that this path is a waste of time (all of the previous results are useful). Are your results *general*, potentially generalizable, or specific to a particular case?

Other Considerations

An abstract must be a fully self-contained, capsule description of the paper. It can't assume (or attempt to provoke) the reader into flipping through looking for an explanation of what is meant by some vague statement. It must make sense all by itself. Some points to consider include:

- Meet the word count limitation. If your abstract runs too long, either it will be rejected or someone will take a chainsaw to it to get it down to size. Your purposes will be better served by doing the difficult task of cutting yourself, rather than leaving it to someone else who might be more interested in meeting size restrictions than in representing your efforts in the best possible manner. An abstract word limit of 150 to 200 words is common.
- Any major restrictions or limitations on the results should be stated, if only by using "weasel-words" such as "might", "could", "may", and "seem".
- Think of a half-dozen search phrases and keywords that people looking for your work might use. Be sure that those exact phrases appear in your abstract, so that they will turn up at the top of a search result listing.
- Usually the context of a paper is set by the publication it appears in (for example, *IEEE Computer* magazine's articles are generally about computer technology). But, if your paper appears in a somewhat un-traditional venue, be sure to include in the problem statement the domain or topic area that it is really applicable to.
- Some publications request "keywords". These have two purposes. They are used to facilitate keyword index searches, which are greatly reduced in importance now that on-line abstract text searching is commonly used. However, they are also used to assign papers to review committees or editors, which can be extremely important to your fate. So make sure that the keywords you pick make assigning your paper to a review category obvious (for example, if there is a list of conference topics, use your chosen topic area as one of the keyword tuples).

Conclusion

Writing an efficient abstract is hard work, but will repay you with increased impact on the world by enticing people to read your publications. Make sure that all the components of a good abstract are included in the next one you write.

Further Reading

Michaelson, Herbert, *How to Write & Publish Engineering Papers and Reports*, Oryx Press, 1990. Chapter 6 discusses abstracts.

Cremmins, Edward, *The Art of Abstracting 2nd Edition*, Info Resources Press, April 1996. This is an entire book about abstracting, written primarily for professional abstractors.

Embedded system designers may be interested in my blog.