Introspective Reasoning:

Anonymous ACL submission

Abstract

As the strong generative capability exploited by Chain of Thoughts (CoT) and amplified by various test-time decoding algorithms, Large Language Models (LLMs) can now reason like a human scholar, at the cost of intensive computation on redundant and repetitive reasoning process based on natural languages. This increasing utilization of CoT can potentially lead to an increasing divergence between what model "think" and "talk". It remains unclear to what extent model's "thinking" diverges from what it "says" when handling all kinds of reasoning tasks. This paper aims to address this question with a comprehensive analysis. Specifically, we explore the following questions: (1) What level of granularity of reasoning does model internally precept itself best? (2) How far can model be aware of its future mistakes or failures? (3) Are there any internal features that can be perceived and modified during model reasoning to circumvent the mistakes before "saying it out loud"? Codes regarding our analytical methods and findings are available at:https://github.com/YourSaDady/Introspective_Reasoning.git

1 Introduction

In recent years, large language models have demonstrated strong capabilities in generation tasks and have already been applied to various fields. Among these, math reasoning stands out as large language models cannot handle as effectively as they do for other traditional NLP tasks. The math reasoning task is characterized by strong causal correlations between consecutive steps and requires both local and global attention to effectively plan for exploration and exploitation. Consequently, it is essential to verify each step to ensure its correctness. Recently, as more tasks aim to test a model's capability to understand the world concept and the logic behind questions, language models play a fundamental role in decision-making. They are enhanced with various decoding methods to aid in model planning and reasoning beyond the traditional generation workflow in NLP. However, to our knowledge, most decoding methods require high computational costs during the fine-tuning stage, if any, and high spatial and temporal costs during the inference stage. Examples include methods based on Monte Carlo Tree Search (MCTS), beam search, or Reinforcement Learning. While these methods are expected to trade-off for better performances in various math reasoning and agent tasks, they often end up distilling the oriented tasks to the language space instead of leveraging the model's internal capabilities for reasoning and planning.

Our experiments show that primarily treating model uncertainty, specifically the entropy from each step, as reward metrics in the current decoding methods does not promise significant performance improvement. Given the drawbacks in current model decoding methods mentioned above, we propose a novel method called CRYptical Casual cHaIn DeCoding (CRYCHIC). Inspired by FactCheckMate, CRYCHIC uses a learned classifier to interpret a reward signal from the base model's hidden states for each step to determine whether the current planning trajectory is leading to an undesirable outcome. Depending on this signal, CRYCHIC leverages an intervention module to adjust the hidden states of problematic steps to minimize the total epistemic uncertainty from the model.

To our knowledge, CRYCHIC is the first method to combine model reasoning with model explainability in a probing-intervention style, and is more efficient compared to the traditional reasoning methods for less parameter to be learned and less inferencing time due to a single model generation. Experimental results show that our classifier can effectively distinguish good and bad reasoning steps with an accuracy of over 80% on GSM8K and MATH, and our intervener (...).

Our contributions are manifold:

1. We find that when dealing with math reasoning tasks, models' earlier steps exhibit a negative correlation between entropy-based uncertainty and progress rate. However, this phenomenon disappears during the later steps of planning.

2. We find that trivial uncertainty-driven decoding methods, such as simply replacing the reward metrics in existing decoding methods like MPC with step-level entropy gradient drop, result in minor performance improvements.

3. We propose a novel decoding method, CRYCHIC, which is (...expected to achieve comparable or better task completeness and shorter inference times on math reasoning tasks compared to current strong baselines.)

4. We give a comprehensive survey on model's internal expressiveness of reasoning steps by probing from different layers with various probe types, giving a quantitative measure and suggestions for future studies.

2 Method

In this session, we first introduce the two probes used for classification and intervention in sections 2.1 and 2.2 respectively. The classifier converts step-wise hidden states from the model's reasoning step to a reasoning quality score, and the intervention module changes the hidden states according to this score. Intuitively, these two probes performs projections between the model's hidden space and the linear score space based on contrastive datasets for reasoning tasks. In section 2.3, we introduce our decoding method CRYCHIC, which simply combines the classifier and the intervention module for inference, and compare the reasoning performance with reasoning baselines, such as MCTS and MPC. For the last section 3.1, we tested all combinations of classification variables as mentioned in 2.1 and summarize our results in Figure **??**.

2.1 Classifier

Definition Suppose S is the total sequence length of the question and all previous steps in the prompt, H is the dimension of the hidden space, we define our classifier $f_\theta \in \mathbb{R}^{H\times 1}$ as

$$f_\theta^l(\mathbf{h}_{I+O}^l) := \sigma(probe(\mathbf{h}_{I+O}^l)) \tag{1}$$

where the input $\mathbf{h}_{I+O}^l \in \mathbb{R}^{S\times H}$ is the sequence of hidden states $(h_{<s,1}^l, h_{<s,2}^l, ..., h_{<s,m}^l)$ corresponds to the question prompt and all the previous reasoning steps until current step s (step s is of length m) on layer l, and $\sigma(\cdot)$ is the sigmoid function. For the choice of $probe(\cdot)$, we tested linear (a single linear projection), non-linear (two linear projection inserted with ReLU as nonlinear), LSTM (with various numbers of layers) classifiers. Since the linear and non-linear classifiers cannot take in the temporal information S, we apply various aggregation methods of hidden states before putting into the probe: mean, last-token and learnable weights.

Dataset To train the classifiers above, we have to obtain samples in the form of $(\mathbf{h}_{I+O,i,j}^l, \alpha_{i,j}^l)$, where the input $\mathbf{h}_{I+O,i,j}^l$ is the sequence of hidden states for step j for solving question i from model's layer l, and the $\alpha \in \mathbb{R}$ is a score representing the rating of how well the future steps can lead to the correct answer. Existing datasets containing rated reasoning steps includes MATH-Shepherd and PRM800K. We prompt our base model with few-shots (8 by default), the question, and all the steps until the current step j and let the model to complete the rest steps to reach a final answer, then we extract the hidden states corresponding to the question and all the given prompts as $\mathbf{h}_{I+O,i,j}^l$.

Since the classifier is simple to evaluate given the sample labels, we divide the dataset into training and validation sets with a proportion of 4:1, and evaluate the trained classifier via the latter.

Training

$$\mathcal{L}_{cls} := BCE(f_\theta(\mathbf{h}_{I+O,i,j}^l), \alpha_{i,j}) \tag{2}$$

We choose Binary Cross Entropy (BCE) as the loss function for training, with a learning rate of 0.001. We use the same setting for the probing survey in 3.1.

2.2 Intervention Module

Definition Similarly, we define our intervention module $g_\phi \in \mathbb{R}^{H\times H}$ as

$$g_\phi^l(\mathbf{h}_s^l) := \sigma(W(LSTM(\mathbf{h}_s^l))) \tag{3}$$

where the input $\mathbf{h}_s^l \in \mathbb{R}^{m\times H}$ is a sequence of hidden states $(h_{s,1}^l, h_{s,2}^l, ..., h_{s,m}^l)$ corresponding to the current step (clue) s of length m on layer l. Taking account of the considerable parameters inside the intervention module due the the output space of $\mathbb{R}^H$, we use a single layer LSTM $LSTM(\cdot) \in \mathbb{R}^{128\times H}$ with output dimension of 128 together with a linear projection $W \in \mathbb{R}^{H\times 128}$

to first convert the step-wise hidden states into a subspace $\mathbb{R}^{128}$ and then project back to the original hidden space $\mathbb{R}^{H}$.

Dataset Similarly, to train the intervention module defined as above, we collect sample pairs in the form of $(\mathbf{h}^l_{chosen,s}, \mathbf{h}^l_{rejected,s})$, where $\mathbf{h}^l_{chosen,s}$ is the sequence of step-wise hidden states corresponding to a step s more likely to lead to the correct answer, and vise versa for $\mathbf{h}^l_{rejected,s}$. To get these hidden states sequences, first we use the existing contrastive reasoning datasets like PRM800K, where each sample contains a pair of chosen and rejected solutions to the same question. We select the samples where the chosen and the rejected solutions have the same number of reasoning steps. Then to obtain the sequence of hidden states corresponding to step s, we prompt the base model with the question, n-shot examples and partial solution steps until step s.

Since it is not intuitive to evaluate the intervention module based on the contrastive dataset, we use all samples for training.

Training

$$\mathcal{L}_{itv} := MSE(g^l_\phi(\mathbf{h}^l_{chosen,s}), \mathbf{h}^l_{rejected,s}) \quad (4)$$

We choose Mean Square Error (MSE) as the loss function for training, with a learning rate of 0.001.

Note that the $\mathbf{h}^l_{chosen,s}$ and $\mathbf{h}^l_{rejected,s}$ are often of different token lengths. To perform token-to-token corresponding intervention to fit the implementation, we add zero paddings to both sequences to a fixed length of 100.

2.3 CRYCHIC

We propose the CRYCHIC decoding pipeline, which simply combines the classifier f_θ and the intervention module g_ϕ we trained previously.

Specifically, inside each ordinary model forward call, the generated decoding token h_i is first processed by the classifier f_θ together with all the previous tokens $\mathbf{h}_{<in-shot}$ corresponding to the input question, the instruction and the previous partial reasoning steps excluding the n-shot examples. At the same time, h_i i also processed by the intervention module g_ϕ together with all the generated decoding tokens $\mathbf{h}_O$ corresponding to the previous steps.

The classifier f_θ generates a signal $\alpha_i \in [0, 1]$, which is served as a multiplier to the intervention vector generated by the intervention module $\Delta h_i := g_\phi(h_i)$.

The ultimate intervened vector h'_i is calculated by

$$h'_i := h_i + \sigma(log(\alpha_i \Delta h_i)) = h_i + \sigma(log(f_\theta(\mathbf{h}_{<i \setminus n-shot}) g_\phi(\mathbf{h}_O))) \quad (5)$$

Note that in 2.1 and 2.2, we train the classifier and the intervention module on a step-level, whereas during inference, the classification and intervention is implemented on a token-level, due to the autoregressive nature of the HuggingFace model's token-wise forwarding.

3 Understanding model introspection of reasoning steps from internal representations

In this section, we look into model's internal activations expecting to capture useful features indicating model's internal behaviors during reasoning. In 3.1, we focus on the output hidden states of the reasoning steps from the transformer layers. We examine all layers and different combinations of probe types and aggregation methods. we also examine the extent to which different ranges of the signal generated by the classifier have on the intervention and thus the overall effect. In 3.2, we utilize the logprobes from model's last output layer to calculate the mean token entropy (MCE) of the reasoning steps, trying to understand the model uncertainty from the view of information theory. Then, we try to utilize this uncertainty signal in reasoning by treating MCE as the reward score in MPC.

3.1 Survey on introspective signal derived from hidden states

subcaption

3.2 Survey on entropy-based uncertainty derived from logits

3.3 Footnotes

Footnotes are inserted with the `\footnote` command.[1]

3.4 Tables and figures

See Table 3 for an example of a table and its caption.
Do not override the default caption sizes.

As much as possible, fonts in figures should conform to the document fonts. See Figure 2 for an example of a figure and its caption.

[1]This is a footnote.

Probe Type	Current Step	+1	+2	+3
Linear (mean, specified)	64.92	64.47	63.74	62.54
Non-Linear (mean, specified)	62.32	63.73	61.68	61.54
LSTM-1-layer (specified)	58.65	68.52	45.95	60.75
LSTM-3-layer (specified)	70.84	66.90	57.68	61.86
Linear (mean, freezed)	?	?	?	?
Non-Linear (mean, freezed)	61.37	59.26	58.19	57.53
LSTM-1-layer (freezed)	58.65	54.7	51.15	48.46
LSTM-3-layer (freezed)	70.84	63.93	61.32	60.18

Table 1: The probing accuracy for predicting correctness on a step-level. I+O represents overall correctness prediction. +1, +2, and +3 represents predicting the correctness of the future steps, specifically, the next, the second next and the third next steps. All probes are trained from the 14th layer's output hiddens of Llama3.1-8B-Instruct.

Metod	Model	GSM8K	MATH	PRM800K	ProcessBench
Pretrained	Llama3.1-8B-Instruct	?	35	?	?
	Qwen2.5-Math-7B	?	?	?	?
	DeepSeek-R1-Distill-Llama-8B	?	?	?	?
	Llama3.1-70B	?	?	?	?
LoRA	Llama3.1-8B-Instruct	?	?	?	?
	Qwen2.5-Math-7B	?	?	?	?
	DeepSeek-R1-Distill-Llama-8B	?	?	?	?
	Llama3.1-70B	?	?	?	?
PEFT_{itv}	Llama3.1-8B-Instruct	?	33	?	?
	Qwen2.5-Math-7B	?	?	?	?
	DeepSeek-R1-Distill-Llama-8B	?	?	?	?
	Llama3.1-70B	?	?	?	?
CRYCHIC	Llama3.1-8B-Instruct	?	?	?	?
	Qwen2.5-Math-7B	?	?	?	?
	DeepSeek-R1-Distill-Llama-8B	?	?	?	?
	Llama3.1-70B	?	?	?	?

Table 2: Results for models with different tuning and decoding methods on reasoning tasks. PEFT_{itv} trains an intervention module outside of the model based on its hidden states before and after the intervention. This is the same as CRYCHIC without the classifier.

Command	Output	Command	Output
{\"a}	ä	{\c c}	ç
{\^e}	ê	{\u g}	ğ
{\`i}	ì	{\l}	ł
{\.I}	İ	{\~n}	ñ
{\o}	ø	{\H o}	ő
{\'u}	ú	{\v r}	ř
{\aa}	å	{\ss}	ß

Table 3: Example commands for accented characters, to be used in, *e.g.*, BibTEX entries.

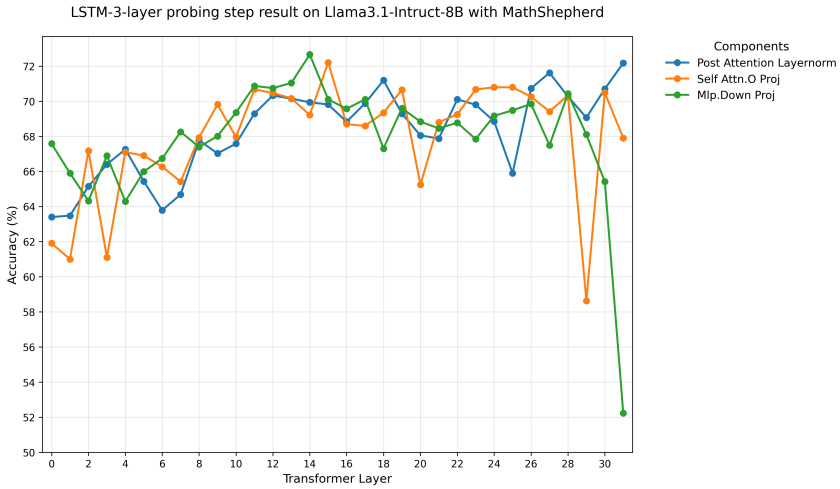
Using the `graphicx` package graphics files can be included within figure environment at an appropriate point within the text. The `graphicx` package supports various optional arguments to control the appearance of the figure. You must include it explicitly in the LaTeX preamble (after the `\documentclass` declaration and before `\begin{document}`) using `\usepackage{graphicx}`.

3.5 Hyperlinks

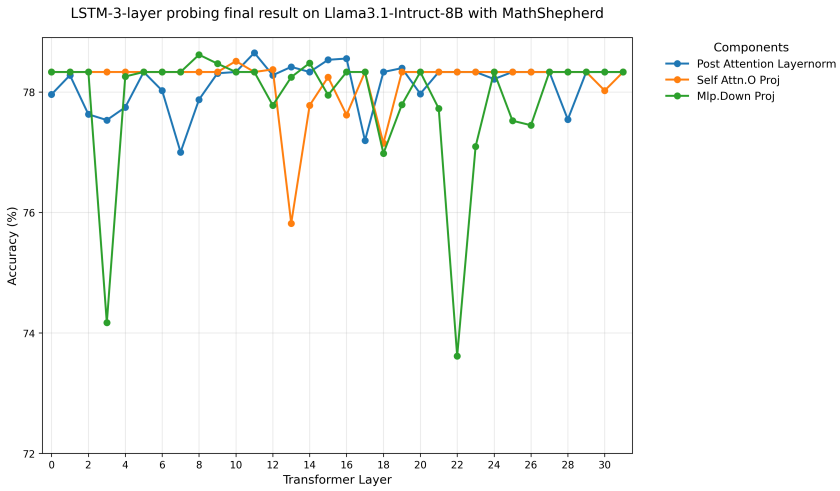
Users of older versions of LaTeX may encounter the following error during compilation:

> `\pdfendlink` ended up in different nesting level than `\pdfstartlink`.

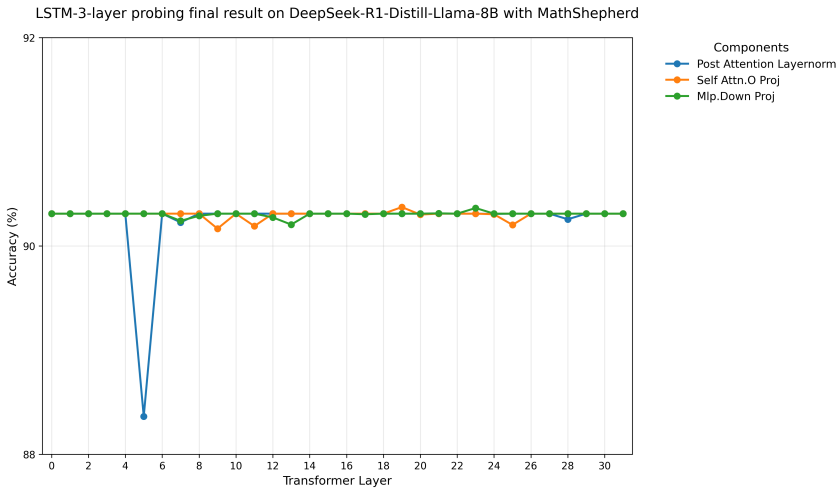
This happens when pdfLaTeX is used and a citation



(a) Accuracies for classifying reasoning steps probing from Llama3.1-8B-Instruct.



(b) Accuracies for predicting final answers probing from Llama3.1-8B-Instruct.



(c) Accuracies for predicting final answers probing from DeepSeek-R1-Distill-Llama-8B.

Figure 1: Probing results across different model layers and tasks

splits across a page boundary. The best way to fix this is to upgrade LaTeX to 2018-12-01 or later.

Output	natbib command	ACL only command
(Gusfield, 1997)	\citep	
Gusfield, 1997	\citealp	
Gusfield (1997)	\citet	
(1997)	\citeyearpar	
Gusfield's (1997)		\citeposs

Table 4: Citation commands supported by the style file. The style is based on the natbib package and supports all natbib citation commands. It also supports commands defined in previous ACL style files for compatibility.

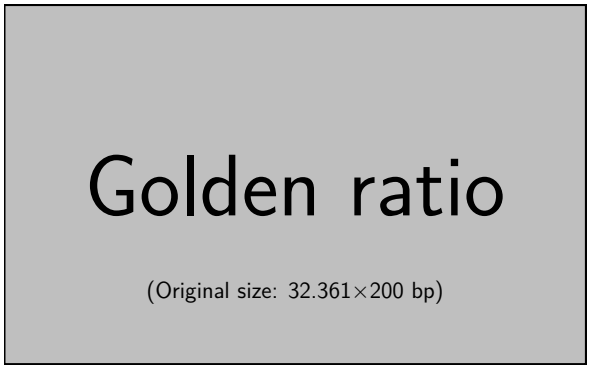


Figure 2: A figure with a caption that runs for more than one line. Example image is usually available through the mwe package without even mentioning it in the preamble.

3.6 Citations

Table 4 shows the syntax supported by the style files. We encourage you to use the natbib styles. You can use the command \citet (cite in text) to get "author (year)" citations, like this citation to a paper by Gusfield (1997). You can use the command \citep (cite in parentheses) to get "(author, year)" citations (Gusfield, 1997). You can use the command \citealp (alternative cite without parentheses) to get "author, year" citations, which is useful for using citations within parentheses (e.g. Gusfield, 1997).

A possessive citation can be made with the command \citeposs. This is not a standard natbib command, so it is generally not compatible with other style files.

3.7 References

The LaTeX and BibTeX style files provided roughly follow the American Psychological Association format. If your own bib file is named custom.bib, then placing the following before any appendices in your LaTeX file will generate the references section for you:

```
\bibliography{custom}
```

You can obtain the complete ACL Anthology as a BibTeX file from https://aclweb.org/anthology/anthology.bib.gz. To include both the Anthology and your own .bib file, use the following instead of the above.

```
\bibliography{anthology,custom}
```

Please see Section 4 for information on preparing BibTeX files.

3.8 Equations

An example equation is shown below:

$$A = \pi r^2 \tag{6}$$

Labels for equation numbers, sections, subsections, figures and tables are all defined with the \label{label} command and cross references to them are made with the \ref{label} command.

This an example cross-reference to Equation 6.

3.9 Appendices

Use \appendix before any appendix section to switch the section numbering over to letters. See Appendix A for an example.

4 BibTeX Files

Unicode cannot be used in BibTeX entries, and some ways of typing special characters can disrupt BibTeX's alphabetization. The recommended way of typing special characters is shown in Table 3.

Please ensure that BibTeX records contain DOIs or URLs when possible, and for all the ACL materials that you reference. Use the doi field for DOIs and the url field for URLs. If a BibTeX entry has a URL or DOI field, the paper title in the references section will appear as a hyperlink to the paper, using the hyperref LaTeX package.

Limitations

Since December 2023, a "Limitations" section has been required for all papers submitted to ACL

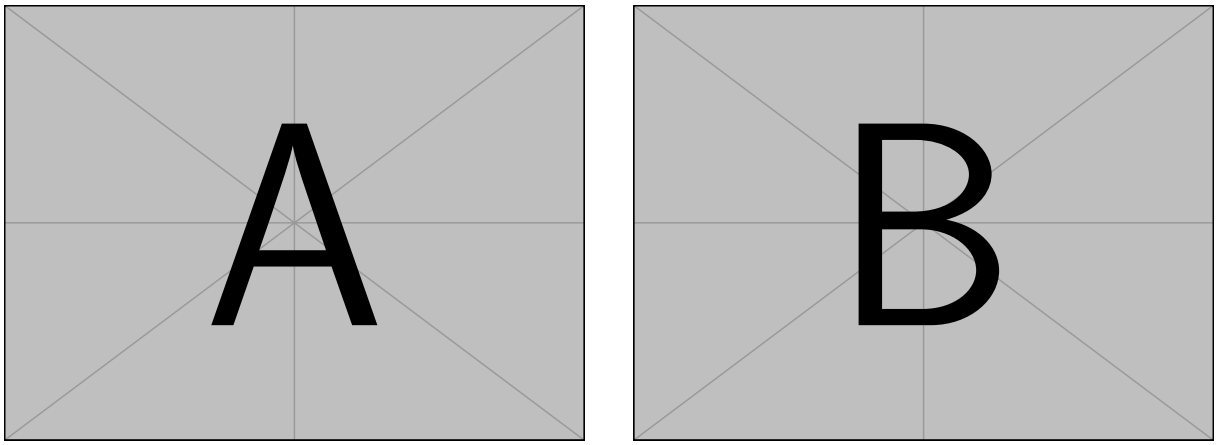


Figure 3: A minimal working example to demonstrate how to place two images side-by-side.

Rolling Review (ARR). This section should be placed at the end of the paper, before the references. The "Limitations" section (along with, optionally, a section for ethical considerations) may be up to one page and will not count toward the final page limit. Note that these files may be used by venues that do not rely on ARR so it is recommended to verify the requirement of a "Limitations" section and other criteria with the venue in question.

Acknowledgments

This document has been adapted by Steven Bethard, Ryan Cotterell and Rui Yan from the instructions for earlier ACL and NAACL proceedings, including those for ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTeX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

A Example Appendix

This is an appendix.