Probabilistic Programming for Voucher Information Extraction

Preliminary Practical Experiences

Ahmad Salim Al-Sibahi
University of Copenhagen/Skanned
ahmad@{di.ku.dk,scanned.com}

Thomas W. Hamelryck
University of Copenhagen
thamelry@binf.ku.dk

Fritz Henglein
University of Copenhagen
henglein@diku.dk

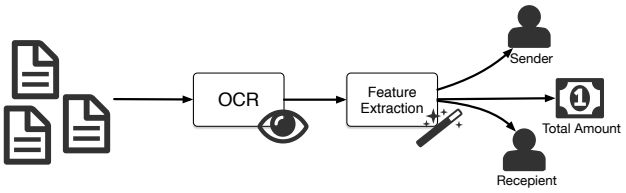

Figure 1. Skanned's Voucher Scanning Pipeline

1 Introduction

Skanned is a service provider whose main service is to provide other companies integration with its Voucher Scanning (VS) system. The VS system (fig. 1) allows extracting information—sender, receiver, amounts—from accounting documents like receipts, invoices, and credit notes. Such task is inherently amendable to probabilistic reasoning: vouchers vary heavily in layout and content, and it is not possible in general to determinstically cover all case while preserving accuracy. Concretely, we will in this paper discuss:

- Choice and preliminary user experiences with existing probabilistic programming frameworks
- Two concrete challenges in Skanned's Voucher Scanning system
- Proposed probabilistic models for the presented challenges and preliminary results

2 Skanned's Voucher Scanner

The core pipeline of Skanned's VS system is intuitively simple. Integration partners send their vouchers through a web service to be processed by the VS pipeline (presented in fig. 1). The pipeline then performs two steps:

- It uses Optical Character Recognition (OCR) to convert Adobe PDF files and images to text boxes.
- It uses Feature Extractors (FEs) to extract information, e.g., sender and total amount, from the OCR text boxes.

The current FEs work completely deterministically using heurestics with parameters determined by the VS system's developers using their domain knowledge. This already provides good results, achieving an 85% accuracy in average across all features without further post-processing. [1], There is room for improvement to cover the remaining 15% by improving on its drawbacks:

Staticness The current system's parameters are fixed by the developers and do not dynamically take into account live data and customize the system for different individual clients.

Confidence The current system does not provide a confidence score on the accuracy of the extracted information. Such score is important to flag for clients performing manual corrections of the extracted results.

3 Framework Selection and Experiences

While probabilistic programming is an emerging field, there have already been multiple powerful frameworks developed across a variety of languages. Skanned's pipeline is written in Python, which made us focus on Python frameworks for the sake of interoperability. Luckily, there are plenty of actively developed Python frameworks for probabilistic programming, including PyMC3 [10], Edward [12], Pyro [13] and ZhuSuan [11]. We will now further discuss aspects of these frameworks that influence our selection, and our preliminary experiences on using their touted features.

Inference Techniques Exact inference of the posterior distribution is intractable for most realistic probabilistic models [3, Part III]. There are two popular types of approximate techniques: *sampling* and *variational inference*.

Sampling All discussed frameworks support modern efficient sampling techniques like Hamiltonian Monte Carlo (HMC) [5, 7] which uses the gradient to direct sampling. Our preliminary experiences show us that sampling is relatively straightforward to use and provides accurate results when successful. Unfortunately, we found it hard to scale to even modestly sized data sets (around 100 vouchers), and using discrete latent features—which we found useful in some models we tried—requires us to fallback to much slower traditional MCMC-based sampling [1].

Variational Inference Modern variational inference [4, 6, 8] supported by the discussed frameworks, can be effectively

PROBPROG '18, October 04–06, 2018, Boston, MA, USA
2018.

[1] The VS system achieves better results in practice, since it internally creates *templates* from vouchers that have already been seen and validated, but this does not work well with unseen voucher types.

combined with deep learning techniques for scalability. Our preliminary experiences show that this technique scales much better to large datasets since it allows mini-batch training, but in turn require a more elaborate set-up compared to sampling. In particular, the choice of the approximating distribution—whose distance should be minimized to the true posterior—is harder for multi-modal distributions and non-convexity makes it necessary to use considerable time tuning hyperparameters to get an acceptable solution.

Business Criteria Our intention is to integrate probabilistic programming in an industrial system and so it is important to consider business-relevant criteria as well.
Source Code Availability. Luckily all the aforementioned frameworks have freely available source code, that is released under an open source license.
Framework Maturity. PyMC3 is a completely re-engineered version of the PyMC framework, which has been developed since 2003. In our preliminary experiences, it is also the framework with widest support of distributions and simplest to use interface in terms of initialization, automation and tuning. Other frameworks are still relatively young. Edward is around 2 years old (from 2016), but has recently been integrated into the main TensorFlow project. Both Pyro and ZhuSuan have been developed since late last year and are still in early stage.
Industrial Support. All frameworks except ZhuSuan, have been sponsored by industrial partners. PyMC3 is sponsored by NumFocus and Quantopian. Edward is developed by Google Brain team that is also behind TensorFlow. Pyro is developed by Uber's AI team.

GPU Support Most of the frameworks tout CUDA support as a feature to accelerate inference for large data set. We set out to test this on a 10-core Intel Xeon CPU and 2 Nvidia Quadro P5000 GPUs. Unfortunately, we had a hard time to exploit GPU performance for inference in our models (using PyMC3 or Pyro), where the powerful CPU usually was quicker.

Choice of Framework Given the stated criteria, our main focus has been using PyMC3. We however found it easy to port our models to other frameworks, and have done so occasionally when we wanted to compare aspects like performance and expressibility.

4 Challenges

We will discuss two concrete challenges in the VS system: i) the FE component for finding the total amount on a receipt, and ii) the algorithm for automated detection of keywords on a voucher.

Finding Total Amounts The FE that is used to find total amounts currently uses a combination of deterministic techniques and heuristics:

1. It looks for a numeric-like value, since amounts are almost always numerical.
2. It tries to prioritize values that are close to relevant keywords, e.g. 'Total' or 'Amount'.
3. It prioritized large amounts over lower ones, since the total amount is usually a sum of other amounts.

The first point is relatively standard, since it simply filters everything that does not look like an amount. Our goal is therefore to probabilistically model the keyword-based prioritization and magnitude. For the keyword-based prioritization, our hypothesis is that more a feature is usually aligned vertically or horizontally to relevant keywords (around 0° or 90°) and the closer it is the better. For the amount value, the larger the relative magnitude (normalized compared to others) the more likely it is the total amount.

We have implemented a simple probabilistic model in PyMC3: we rely on the position to the closest positive keyword, using a using a mixture model for the angle (with means 0 and $\frac{\pi}{2}$ radians) and skewed normal distributions for distances and relative magnitude. For a given voucher we then choose the amount that has the largest probability according to the model. This has so-far worked well, correctly identifying around 80% of the total amounts in our test set. Our plan is now to generalize the model to increase accuracy, by using a Bayesian non-parametric model that takes into account a variable number of keywords and other potentially interesting features (e.g., absolute position on page and page number).

Keyword Detection Currently, the VS system uses a manually pre-compiled list of keywords to guide the information extraction process. There is however an interest in automatically detecting new keywords, since the current system is not scalable and hard to adapt for new languages. Our proposed solution is as follows:

1. Rely on DF-ITF [9] to find words that are commonly occuring across vouchers, but rate in a particular voucher. This produces a set of proposed keywords.
2. Identify which features a particular proposed keyword is relevant to, using existing validated data set of found features. We rely on the same positional assumptions as in total amount localization but flip the problem: given we know where our target feature is, how relevant does a particular proposed keyword seem?

For the first point, we used a simple deterministic calculation since we were only interested in proposals and confidence is here less important.

For the second point we are aiming to use a probabilistic model. We are currently in an ongoing effort in evaluating different models for the task. One of our models is inspired by existing Gaussian [2] and Spatial [14] LDA techniques, and tries to cluster potential keywords around different features using the distances and angles to them. Our other model

relies on Beta-Bernoulli latent feature modelling, and tries to infer the relevance of a keyword to each feature directly using the same positional information. We have implemented both models and we are in the process of tuning the training of them; our results so-far have been inconclusive.

References

[1] Siddhartha Chib and Edward Greenberg. 1995. Understanding the metropolis-hastings algorithm. *The american statistician* 49, 4 (1995), 327–335.

[2] Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for Topic Models with Word Embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. The Association for Computer Linguistics, 795–804. http://aclweb.org/anthology/P/P15/P15-1077.pdf

[3] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2014. *Bayesian Data Analysis* (3 ed.). Chapman & Hall/CRC.

[4] Matthew D. Hoffman, David M. Blei, Chong Wang, and John William Paisley. 2013. Stochastic variational inference. *Journal of Machine Learning Research* 14, 1 (2013), 1303–1347. http://dl.acm.org/citation.cfm?id=2502622

[5] Matthew D. Hoffman and Andrew Gelman. 2014. The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research* 15, 1 (2014), 1593–1623. http://dl.acm.org/citation.cfm?id=2638586

[6] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. 2017. Automatic Differentiation Variational Inference. *Journal of Machine Learning Research* 18 (2017), 14:1–14:45. http://jmlr.org/papers/v18/16-107.html

[7] Radford M. Neal. 2011. MCMC using Hamiltonian dynamics. Chapman & Hall / CRC Press, Chapter 5.

[8] Rajesh Ranganath, Sean Gerrish, and David M. Blei. 2014. Black Box Variational Inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics, AISTATS 2014, Reykjavik, Iceland, April 22-25, 2014*. 814–822. http://jmlr.org/proceedings/papers/v33/ranganath14.html

[9] Marçal Rusiñol, Tayeb Benkhelfallah, and Vincent Poulain D'Andecy. 2013. Field Extraction from Administrative Documents by Incremental Structural Templates. In *12th International Conference on Document Analysis and Recognition, ICDAR 2013, Washington, DC, USA, August 25-28, 2013*. IEEE Computer Society, 1100–1104. https://doi.org/10.1109/ICDAR.2013.223

[10] John Salvatier, Thomas V. Wiecki, and Christopher Fonnesbeck. 2016. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science* 2 (2016), e55. https://doi.org/10.7717/peerj-cs.55

[11] Jiaxin Shi, Jianfei. Chen, Jun Zhu, Shengyang Sun, Yucen Luo, Yihong Gu, and Yuhao Zhou. 2017. ZhuSuan: A Library for Bayesian Deep Learning. *arXiv preprint arXiv:1709.05870* (2017).

[12] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. 2016. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787* (2016).

[13] Uber AI Labs. [n. d.]. Pyro: Probabilistic Programming Language. http://pyro.ai/

[14] Xiaogang Wang and Eric Grimson. 2007. Spatial Latent Dirichlet Allocation. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis (Eds.). Curran Associates, Inc., 1577–1584. http://papers.nips.cc/paper/3278-spatial-latent-dirichlet-allocation

Acknowledgments

We would like to thank Dan Rose and the VS developers at Skanned (Bjørn Kaae and Toke Reines) for providing feedback and valuable domain knowledge for this work. This material is based upon work supported by the Innovation Fund Denmark under Grant No. 7039-0072B. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the funding agencies.