International Journal of Scientific Research & Engineering Trends
Volume 6, Issue 4, July-Aug-2020, ISSN (Online): 2395-566X

A Survey on Sales force and AI

Sarthak Ghosh
Department of Computer Science and Engineering
University, India
sanjibghosh2012@gmail.com

***Abstract* - Salesforce is very demanding cloud computing technology in IT industry, which is available on cloud, no need install any software as well as no hardware required. Salesforce.com (SFDC) is currently number one on demand CRM, which runs on force.com platform, as well as CRM is a model used to manage organization interactions like phone calls, Emails, Meetings, issues, Social media with customers and also prospects penetrating to Sales, Marketing and Support. In this paper, we are discussing about Introduction to Cloud Computing, Service models in Cloud Computing, Types of Cloud Computing, Architecture of Cloud Computing and Introduction to MVC as well as SFDC MVC. Further discussing about Introduction to Salesforce, SOQL,Apex and Its Comparison Operators and at last covering Force.com IDE and CRM. The aim of this paper is to show mainly importance of Salesforce.com which is a software giant that manages to give the buyer a easy to use as well as extremely effective CRM solution.Salesforce Research has made significant progress in deep learning and natural language processing (NLP) over the course of the last year that address these challenges and also make good spot in AI. We've built faster, more scalable model architectures and stable model, developed a reinforcement learning agent that programs new neural network architectures, and improved training methods in order to take full advantage of the huge of data currently available and to increase model performance for each individual NLP task. Today, Salesforce excited to announce new breakthroughs that bring us closer to a unified approach to tackling the many facets of natural language, paving the way for humans and machines to communicate more effectively and work side-by-side.**

***Keywords*- Cloud Computing , Marketing and Support, NLP task etc.**

I.INTRODUCTION

Traditional phonetic-based recognition approaches require training of separate components such as pronunciation, acoustic and language model. Since the models are all trained separately with different training objectives, improving one of the components does not necessarily required lead to performance improvement of the whole system. This makes improving of the system performance difficult. End-to-End models address the aforementioned problem by jointly train all components together with a single objective, and thus simplifies the AI model training process significantly.

We tackle these two challenges by
1) Improving the regularization of the model during training period and
2) Using policy learning to optimize directly on the performance metric.

Both approaches are highly effective, powerful and improve the performance of the end-to-end speech model significantly. Without human intuition and insights however, automated architecture search is like fraught with peril. Even at the blinding speed computers operate, our build machines aren't fast enough to brute force search randomly through the billions of different potential architectures. While we can't scale human intuition, we can approximate guess it with through a ranking function. This ranking function is a neural network that learns from past experiences regarding which types of neural networks look promising and which are likely to be useless. This kind of organization is very easy and simple to understand because each element of the scene ('person', 'dog', 'next to', etc.) gets its own representation. Capturing the image this way could provide a fairly and clear ideal scaffold for extracting information relevant to answering a given question.

Model learns to count by enumerating relevant objects in the scene, formulated as the process of sequentially deciding which object need to add to the count. We model how the choice to count one object affects the way other objects are used in subsequent steps and train the network using reinforcement learning (RL) [1]. For all our experiments, we represent and describe the image as the set of object proposals generated from a pre-trained vision module. The inputs are the question and the object proposals from the vision module. IRLC uses the question to give each object an initial score, then cycles through picking and taking the highest-scoring object and updating the scores based on the selection. When none of the remaining objects have a high enough score, the process terminates, giving a approx. count equal to the number of selected objects [1].

International Journal of Scientific Research & Engineering Trends
Volume 6, Issue 4, July-Aug-2020, ISSN (Online): 2395-566X

II. RELATED WORK

Author proposes a hierarchical RL approach which can reuse previously learned skills alongside and as subcomponents of new skills. It achieves this by discovering without human input the underlying hierarchical relations between skills. As the RL model executes a given task, it breaks the task into smaller parts and eventually basic actions using a hierarchy of "policy networks," or neural networks that predict actions, operating at different levels of abstraction. To represent the skills and their relations in an interpretable way, we also communicate all tasks to the RL agent using human instructions like as "put down." This allows the agent in turn to communicate its plans and decisions using human language. Below is the process of processing language.

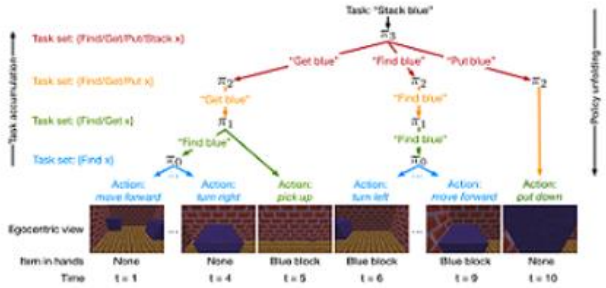

Fig: 1illustrates.

Figure 1 illustrates an example: given the instruction of "Stack blue," which represents a typical complex task, the model executes actions in a multi-level hierarchy to stack two blue blocks together. Steps from the top-level term and policy (i.e., the red branches) outline a learned high-level plan: "Get blue" (find and pick up one blue block) -> "Find blue" (Locate a second blue block) -> "Put blue" (Place the first block on the second block). In addition, the components of this complex plan may themselves have hierarchical structure. Based on an intermediate-level term and policy, for instance, the task "Get blue" has two steps, "Find blue -> action: turn left," whereas "Put blue" can be executed by a single action, "put down.". Tasks accumulate progressively from lower to higher-level policies, while the higher level policy is said to "unfold" into smaller components [10].

The model learns to count by enumerating relevant objects in the scene, formulated as the process of sequentially deciding which object to add to the count. This model how the choice to count one object affects the way other objects are used in subsequent steps and train the network using reinforcement learning (RL). For all our experiments, the author represents the image as the set of object proposals generated from a pre-trained vision module. IRLC in action. The inputs are the questions and the object plan and proposals from the vision module. IRLC uses the questions and observation to give each object an initial score, then cycles through picking the highest-scoring object and updating the scores based on the selection. When none of the remaining other objects have a high enough score, the process terminates, giving a count equal to the number of selected objects [2].

After training happened on counting questions, IRLC outperforms the other baselines we experimented with, including a model adapted from the current state of the art for VQA. In addition, IRLC points to the exact object proposals on which each count is based. In other words, it returns visually-grounded counts. Compared to the attentional focuses produced by a more conventional and perfect model, we argue that the visually-grounded counts of IRLC provide a simpler and more intuitive lens for interpreting model behavior [2].

The author set up our work this way for a few reasons. In particular, previous studies don't train on or report performance on counting specifically, so author need to do those experiments ourselves. In addition, setting the experiments up this way lets system control for many of the little decisions that likely affect absolute performance. For example, by using object proposals from the pre-trained model to serve as the vision module, system ensure that all our counting models use the same visual features.

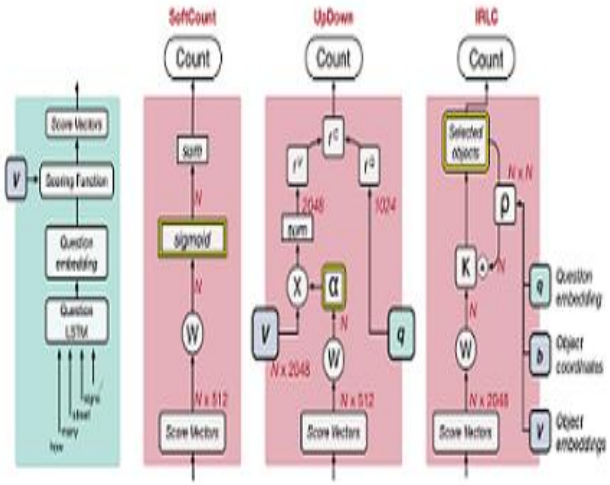

Fig: 2Counting architectures.

Figure 2 Counting architectures. The question module (left) converts the question and object embedding's to score vectors. Each counting module converts the score vectors to counts (number) [2]. Here, N denotes the number of object proposals. The text in red describes the size/dimensionality of the adjacent terms. Boxes with yellow outlines denote to the values used during visualization. Soft Count converts each object into a scalar value and then counts by summing those values. Up Down (modeled after Anderson et al. 2017) produces each object an attention weight, α, and estimates the count from the attention-weighted object encodings and the question encodings. Each bucket labeled with an 'f' indicates a non-linear layer. IRLC generate set of scores, and iteratively selects and object and updates the scores using the interaction matrix, ρ. The interactions are calculated from the variables shown on the right.

Introduce Interpretable Reinforcement Learning Counter--or IRLC for short. With IRLC, the author aim to learn how to count by learning what to count. Author assume that each counting question implicitly refers to a group of objects within the scene that meet some criteria. In this sense, the goal of IRLC is to enumerate each of the objects in that group. This means that, when generating a count, IRLC must choose which specific object proposals it wants to include in the count. The model this process as a sequence of discrete choices.

Importantly, the authors training data doesn't label the objects that are relevant to the question, it only gives us the final answer. So, all we can tell this model during training is how far away the estimated count was from the ground truth. Because of this tight guideline and the discrete nature of the counting choices, we train using RL. Since κ can be interpreted as a part of selection probabilities, the author's approach is well suited for policy gradient methods. We specifically use Self-Critical Sequence Learning, which involves stochastically sampling object selection decisions using the probabilities given by κ and comparing the outcome to what happens if objects are chosen greedily (as at test-time and described above). The network is trained to increase (decrease) the probability of selection sequences that did better (worse) than the greedy selection [3].

Typically, studies on VQA provide performance metrics for each of 3 question categories: 'yes/no', 'number', and 'other'. For our purposes, we need to be more precise, because the 'number' category includes both counting and non-counting types of questions. To deal with this, Salesforce construct a new combination of the training data from the VQA v2 and Visual Genome datasets. We isolate the questions that require counting and whose ground truth answer is in the range of 0-20. For development and testing, we apply the same filter to the validation data from VQA v2. We set aside 5000 of those questions as our test split and use the rest for development (i.e. early stopping, analysis, etc.) [4].

Salesforce call this dataset HowMany-QA. Author trained each of the model variants described above on the training split of How Many-QA. Author rely on 2 statistics to evaluate the counts produced by the models. The first of these metrics is accuracy--which is the standard metric for VQA. This simply tells us how often the best guess of the model was correct, so higher is better. However, accuracy doesn't capture the typical scale of the error, also want to capture how far, on average, each model's answers were from the ground truth. For that, author also calculate the average squared error of the estimated counts and take the square root of that value (root mean squared error, or RMSE). Since RMSE measures error, lower is better. These values are shown the following table:

Model	Accuracy	RMSE
SoftCount	50.2	2.37
UpDown	51.5	2.69
IRLC	56.9	2.39

Fig. 3 Performance Table.

A couple things stand out. First, IRLC is substantially more accurate than the other two baselines (SoftCount and UpDown). Second, the two baselines trade-off accuracy and RMSE. That is, SoftCount returns counts that are on average closer to the ground truth than UpDown (lower RMSE) but are less often exactly correct (lower accuracy). So, these two metrics capture subtle differences in the quality of the counting performance. This underscores the observation that IRLC performs strongly in both metrics [5].

The quantitative performance of IRLC shows that a discrete decision process built on discrete visual representations is not only viable for counting in VQA but superior to alternative methods. But what about our intuition that such an approach should also simplify interpretation of model behavior? There's no clear metric for saying whether one model is more interpretable than another, but Chandrasekaran et al. (2017) have made some progress in bringing this idea into the experimental realm. For now, we'll form a more qualitative argument by borrowing a central idea from their work: that a key hallmark of interpretability is the ability to understand failure modes.

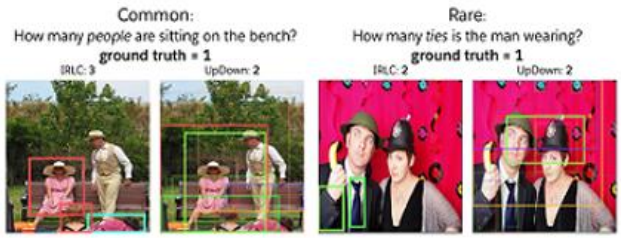

Fig.4 Failure Cases in IRLC

Rather than provide an exhaustive set of examples for some failure mode, The Author illustrate a pair of failure cases that exemplify two trends observed in IRLC; these are shown in Figure 2.2. These two cases also led to failures in the UpDown model, so we include its outputs and attention focuses for comparison. IRLC has little trouble counting people (the most frequently encountered counting subject in the training data), but, as shown in the left example, it encounters difficulty with referring phrases (in this case,

"sitting on the bench") and fails to ignore 2 people in the foreground. In the right example, the question refers to "ties," a subject for which the training data has only a handful of examples. When asked to count ties, IRLC includes an irrelevant object--the man's sleeve. This reflects the broader tendency for IRLC to have difficulty identifying the relevant object proposals when the question refers to a subject that was underrepresented during training. The nature of these failures are obvious by virtue of the grounded counts, which point out exactly which objects IRLC counted. In comparison, the attention focus of UpDown (representing the closest analogy to a grounded output) does not identify any pattern. From the attention weights, it is unclear which scene elements form the basis of the count returned by UpDown [5].

The domain specific language (DSL) defines the search space of potential architectures and allows the computer to compile candidate architectures into executable code. The DSL has two sets: a basic subset and an extended subset. The basic DSL can define the majority of standard RNN architectures (RNN, GRU, LSTM, QRNN ...), suggesting it has can cover much of the existing RNN search space. The extended DSL allows us to incorporate unusual operators like geometric curves, division, or positional encoding. These components are non-standard and haven't been as well explored by humans yet, either due to time or their pre-existing biases as to what does or doesn't work [6].

With the DSL, a standard tanh-RNN may be represented as: tanh(Add(MM(x),MM(h))).

For more complex RNNs, the definition can become a quite a bit more lengthy!
To allow for the architecture search to improve in an automated manner, The author suggest a design a ranking function that estimates a given candidate architecture's performance. Given an architecture-performance pair, the ranking function constructs a recursive neural network that reflects the nodes in a candidate RNN architecture one-to-one. Sources nodes are represented by a learned vector and operators are represented by a learned function. The final vector output then passes through a linear activation and attempts to minimize the difference between the predicted and real performance [7].

Over time the ranking function becomes more and more accurate it predicting a given candidate's performance, meaning it only selects promising architectures to evaluate. The confusion around architecture number 800 is that we allowed the generator to design more complex RNN architectures, specifically those which use both a long and short term memory component.

This suggests the system is able to learn to use diverse and complicated components within RNN architectures. On inspecting the architectures generated, many broke with human intuitions regarding what RNN architectures should or shouldn't work. This suggests that humans may be handicapping their creativity by relying too strongly on ungrounded intuition rather than wide scale experimental results! [8]

Given the differences in generated architectures and their quite varied usage of components, author explored how the models reacted to the same input data over time. Each of the activations differs substantially from those of the other architectures even though they are parsing the same input.

In [10] authors used a new clustering algorithm based on the mutual vote, which adjusts itself automatically to the given dataset, needs minimum overhead in terms of parameters, and also able to detect clusters with different densities in the same dataset. Currently, many Voting/Rating machine searing based automated recommender software systems are developing continuously by many companies for voting based selection of products. A customer would be interested in purchasing products that are similar to the products that he/she liked earlier.

As the input features are likely to not only be captured in different ways but also stored and processed differently, this suggests that ensembles of these highly heterogeneous architectures may be effective. Rather than selecting just the best found architecture, it may become promising to keep an entire zoo of them, selecting and mixing between their results! The resulting architectures do not follow human intuition yet perform well on their targeted tasks, suggesting the space of usable RNN architectures is far larger than previously assumed [9].

Think of AI as an iceberg. What we see as a user is only the tip — but beneath the surface lurks a behemoth support system of data scientists and engineers, massive amounts of data, labor-intensive extraction and preparation and making of that data, and a huge technology infrastructure. It required a specialized team of data scientists and developers to access the correct data, prepare the data, build the correct models, and then integrate the predictions back into an end-user experience such as CRM. Salesforce Proposed Model and Author designed Salesforce Einstein so that all those challenges are ours instead of yours. That means everyone can able to use AI to work smarter in their CRM.

III. CONCLUSION

Today's advancements hold remarkable potential impact for how we move forward as an industry to build a more productive and intuitive relationship between humans and machines. There is still more work to be done, and our focus for the coming year is to continue to push the boundaries of research in deep learning and NLP. The author shows that the performance of the end-to-end speech models can be improved significantly by performing proper regularization and adjustment to the training objective. In particular, the author demonstrates that proper use of data augmentation

and recurrent dropout leads to significant better generalization performance of the model. Additionally, with policy learning, we can further closing the gap between the training objective and the testing metric [9]. Combining all these techniques we achieved a relative performance improvement of ~30% across all datasets over the baseline. The author presents an interpretable approach to counting in visual question answering, based on learning to enumerate objects in a scene. By using RL, Author are able to train the model to make binary decisions about whether a detected object contributes to the final count. Author experiment with two additional baselines and control for variations due to visual representations and for the mechanism of visual-linguistic comparison. Our approach surpasses both baselines in each of our evaluation metrics. In addition, author's model identifies the objects that contribute to each count. These groundings provide traction for identifying the aspects of the task that the model has failed to learn and thereby improve not only performance but also interpretability.

IV. FUTURE WORK

In future we can try other performance measures and other machine learning techniques for better comparison on results. This analysis will help the industries to predict the quality of the different type of scenario based on certain attributes and also it will helpful for them to make good product in the future and make it fully automated and visualized through CRM.

REFERENCES

[1] https://www.salesforce.com/research/
[2] https://blog.einstein.ai/interpretable-counting-for-visual-question-answering/ By: Alex Trott.
[3] https://blog.einstein.ai/improving-end-to-end-speech-recognition-models/ by Yingbo Zhou, CaimingXiong, and RicharSocher 2017.
[4] https://www.salesforce.com/products/einstein/airesearch/research-improving-natural-language-understanding/Richard Socher, Chief Scientist, Salesforce - December 14, 2017.
[5] https://blog.einstein.ai/a-domain-specific-language-for-automated-rnn-architecture-search/ By: Stephen Merity.
[6] https://arxiv.org/abs/1712.07316 – A Flexible Approach to Automated RNN Architecture Generation by Stephen Merity.
[7] https://www.researchgate.net/publication/267695575_Extremely_effective_CRM_Solution_Using_Salesforce.
[8] https://www.salesforce.com/in/products/einstein/ai-deep-dive/
[9] https://www.slideshare.net/NimishChaini/salesforce-research-paper-93972853
[10] https://blog.einstein.ai/thinking-out-loud hierarchical-and-interpretable-multi-task-reinforcement-learning/ Tianmin Shu, Caiming Xiong, and Richard Socher. 2017.