ARES: A Reading Comprehension Ensembling Service

Anthony Ferritto, Lin Pan, Rishav Chakravarti, Salim Roukos
Radu Florian, J. William Murdock, Avirup Sil*
IBM Research AI
Yorktown Heights, NY
aferritto@ibm.com
{panl, rchakravarti, roukos, raduf, murdockj, avi}@us.ibm.com

Abstract

We introduce ARES (A Reading Comprehension Ensembling Service): a novel Machine Reading Comprehension (MRC) demonstration system which utilizes an ensemble of models to increase F1 by 2.3 points. While many of the top leaderboard submissions in popular MRC benchmarks such as the Stanford Question Answering Dataset (SQuAD) and Natural Questions (NQ) use model ensembles, the accompanying papers do not publish their ensembling strategies. In this work, we detail and evaluate various ensembling strategies using the NQ dataset. ARES leverages the CFO (Chakravarti et al., 2019) and ReactJS distributed frameworks to provide a scalable interactive Question Answering experience that capitalizes on the agreement (or lack thereof) between models to improve the answer visualization experience.

1 Introduction

Machine Reading Comprension (MRC) involves computer systems that can take a question and some text and produce an answer to that question using the content in that text. This field has recently received considerable attention, yielding popular leaderboard challenges such as SQuAD (Rajpurkar et al., 2016, 2018) and NQ (Kwiatkowski et al., 2019).

Currently, the top submissions on both the SQuAD and NQ leaderboards combine multiple system outputs. These ensembled systems traditionally outperform single models by 1-4 F-measure. Unfortunately, many of the papers for these systems provide little to no information about the ensembling techniques they use.

In this work, we use GAAMA, a prototype question-answering system using the MRC techniques of (Pan et al., 2019), as our starting point and explore how to ensemble multiple MRC models from GAAMA[1]. We evaluate these techniques on the NQ short answer task. Using our ensemble of models, for each example (question, passage pair), we take the top predictions per system, group by span (answer extracted from the passage), normalize and aggregate the scores, take the mean score across systems for each span, and then take the highest scoring short and long answer spans as our final prediction. These improved ensembling techniques are applied to our MRC systems to produce stronger answers.

Whereas other systems such as (Chakravarti et al., 2019; Yang et al., 2019a) and Allen NLP's[2] make use of a single model, we are able to use multiple models to produce a stronger result. We further take advantage of the fact that both the individual model predictions and the ensembed predictions are returned to help increase explainability for the user. For the graphical interface we use a heatmap to show the level of (dis)agreement between the underlying models along with the "best ensemble" answer. An example of this can be seen in Figure 1.

More completely, our contributions include:

- A novel MRC demonstration system, which leverages multiple underlying MRC model predictions and ensembles them for the user.
- A system architecture that provides scalability to the system designer (by leveraging the cloud ready CFO[3] (Chakravarti et al., 2019) orchestration framework) and flexibility to add and remove models based on the desired latency versus accuracy trade-off.

*Corresponding author.

[1] ARES can use any MRC model.

[2] https://demo.allennlp.org/reading-comprehension

[3] https://github.com/IBM/flow-compiler/

Proceedings of the 2020 EMNLP (Systems Demonstrations), pages 31–37
November 16-20, 2020.

Figure 1: ARES client interface. The correct answer 2018 is boxed and the MRC system answers are highlighted based on a heatmap.

- A GUI with enhanced explainability that allows users to see the (dis)agreement of responses from individual models.
- An analysis of various ensembling strategies with experimental results on the challenging NQ dataset which show that diversity of models is better for ensembling than seed variation. We detail the process for selecting the "best-diverse" set.

2 Related Work

2.1 Ensembled MRC Systems

There have been multiple works creating systems utilizing MRC models. BERTserini (Yang et al., 2019a) is an end-to-end question answering system utilizing a BERT (Devlin et al., 2019) model. (Ma et al., 2019) creates an end-to-end dialogue tracking system featuring an XLNet (Yang et al., 2019b) model. (Qu et al., 2020) performs conversational question answering and utilizes separate ALBERT (Lan et al., 2019) encoders for the question and passage in addition to a BERT (Devlin et al., 2019) model. Allen NLP's MRC demo provides reading comprehension through the use of a variety of different model types. However, to the best of our knowledge we are the first to propose using an ensemble of MRC models to provide a MRC service.

There have likewise been multiple approaches to visualization of system results. BertSerini highlights the answer in the context. Allen NLP's demo allows using gradients to view the most important words in the passage. ARES allows for viewing the most important regions of the passage from the perspective of different models in addition to boxing in the ensembled answer as seen in Figure 1.

2.2 Ensembling Techniques

Many of the top recent MRC systems publish few details on their ensembling strategies. Systems such as (Devlin et al., 2019; Alberti et al., 2019; Liu et al., 2019; Wang et al., 2019; Lan et al., 2019; Group, 2017; Seo et al., 2016) report using ensembles of 5 to 18 models to gain 1.3 - 4 F1 points on tasks such as GLUE, SQuAD 1.0, and SQuAD 2.0; unfortunately most of these systems report little information on their ensembling techniques. (Liu et al., 2020) reports slightly more information: gaining 1.8 and 0.6 F1 points short answer (SA) and long answer (LA) respectively on the NQ dev set with an ensemble of three models with different hyperparameters.

We also consider work in the field of information retrieval (IR) as a way to aggregate multiple scores for the same span. Similar to the popular CombSUM and CombMNZ (Kurland and Culpepper, 2018; Wu, 2012) methods, considering the spans as the "documents", we use span-score weighted aggregation in our noisy-or aggregator. Futher, we additionally incorporate the use of rank-based scoring from Borda (Young, 1974) and RRF (Cormack et al., 2009) for our exponential sum approach (in addition to utilizing score for this approach). We finally consider a reciprocal rank sum aggregation strategy based on the ideas in RRF (Cormack et al., 2009). To our knowledge this is the first published application of IR methods for this purpose.

3 System Overview

We describe the architecture of the system and additionally provide an overview of the client (GUI) used in this demonstration. The system is composed of MRC and ensembling services which are orchestrated by CFO. The MRC services (in our case GAAMA) provide reading comprehension via a transformer model (Pan et al., 2019); multiple services utilizing different model architectures are run to extract answers for a given question and passage. After the MRC services extract their answers, they are all passed to ARES which ensembles the results. The ensembling algorithm used by ARES is detailed in Sections 4 and 5. Note that the MRC service only extracts short answers, therefore only those portions of our ensembling approach are used.

Both the ensembled and original answers are then returned to the caller, allowing the clients to display the final ensembled answers and the original answers they were generated from to the end user.

More completely, the system takes the following as input through a grpc (Talvar, 2016) interface: question, passage, minimum confidence score threshold δ, maximum number of answers N, maximum number of answers per model n, and number of models k. These inputs are sent from the client (we discuss our client below) and received by the CFO node which orchestrates the containers. The choice of k is bounded on how many GAAMA containers are deployed (e.g. if there are 3 then $k \in \{1, 2, 3\}$). By tweaking the parameter k, clients can opt for increased accuracy (higher k) or decreased latency (lower k) as when multiple models run on the same GPU the request latency increases. As depicted in Figure 2 (where there are 3 MRC models running), each of the k=2 GAAMA containers then receive the question and passage from CFO, returning at most n answers to CFO. These answers, together with their confidence scores, are then sent to the ensembler by CFO which produces at most N ensembled answers (each with confidence score at least δ) and returns them to CFO. Finally, both the answers of the k models and the ensembled answers predicted by ARES are returned by CFO to the caller.

The GUI client for our system is based on a ReactJS[4] web interface. A request is taken as input from the user and sent to the system where is is processed as described above. When an answer with sufficient confidence score is returned, it is displayed to the user as seen in Figure 1. Both the ensembled answer and the individual answers are shown together with their respective confidence scores. These answers are also shown in the context of the original passage. The ensembled answer is boxed in. For the individual answers a character heatmap is created representing how many of the candidate answers each character appears in. This heatmap is used to highlight the passage different different colors corresponding to the heatmap (characters not used in any answers are not highlighted). Both the boxing and highlighting of answers are done using MarkJS[5]. Note that while these visualizations only show the top answer for each MRC model, n answers per model are ensembled together. If an answer with sufficient confidence score is not returned, this is relayed to the user through the GUI.

[4] https://reactjs.org/

[5] https://markjs.io/

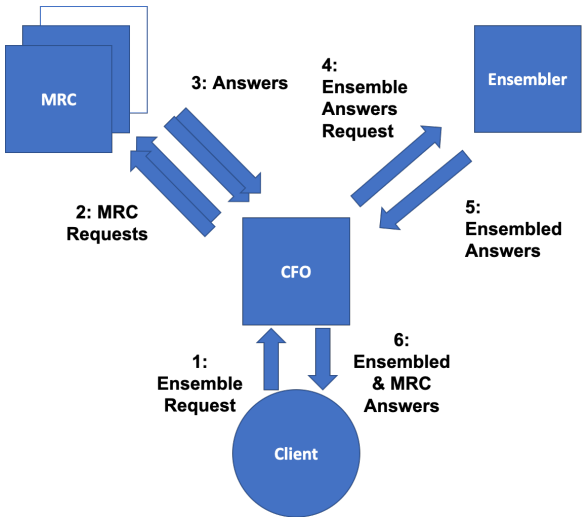

Figure 2: Architecture of the the ARES system. We use GAAMA as our MRC service.

4 Methods

We investigate a number of strategies for ensembling models on the NQ dataset. We use the NQ dataset as it is more realistic and challenging than SQuAD, as its questions were created by Google Search users prior to seeing the answer documents (so they do not suffer from observational bias). In order to formally compare approaches we partition the NQ dev set into "dev-train" and "dev-test" by taking the first three dev files for the "train" set and using the last two for the "test" set (the original dev set for NQ is partitioned into 5 files for distribution). This yields "train" and "test" sets of 4,653 and 3,177 examples respectively.

For each strategy considered we search for the best k-model ensemble over the "train" set and then evaluate on the "test" set. For these experiments we use $k = 4$ as this is the number of models that we can decode in the 24 hours (the limit for the NQ leaderboard). We begin by outlining our core strategy that underlies the approaches we have investigated.

Using this strategy we investigate a baseline approach of ensembling multiple versions of the same model trained with different seeds. We then investigate search strategies for choosing the best models from candidates trained with different hyperparameters, in addition to different normalization and aggregation strategies that are used on a set of candidates.

4.1 Core Strategy

For each example processed by the k systems being ensembled, our system assigns a score to each long and short span according to the normalization and aggregation strategies (see below). Note that our system currently only predicts single short spans rather than sets, so we currently score each short span independently.

We use the top-20 candidate long and short answers (LA and SA respectively) for each system. We experimented with additional values, but empirically found 20 to provide an ideal accuracy versus latency trade-off given hardware resources. To combine systems we take the arithmetic mean of the scores for each long and short span predicted by at least one system. We have experimented with other approaches such as median, geometric mean, and harmonic mean; however these are omitted here as they resulted in much lower scores than arithmetic mean. For spans which are only predicted by some systems a score of zero is assigned (for the systems which do not predict the span) to penalize spans which are only predicted by some systems. The predicted long span is then the span with the greatest arithmetic mean. Similarly for short answers the predicted span is the one with the greatest arithmetic mean (it must also be in a non-null long answer span).

4.2 Seed Ensembles

We first examine the baseline approach of ensembling k versions of the same model trained with the same hyperparameters, only varying the seed between models. We use the model with the best hyperparameters based on (Pan et al., 2019) having the highest sum of short and long answer F1 scores on dev. This model is trained $k-1$ additional times with different seeds and then they are all ensembled using the core strategy.

4.3 Search Strategies

We consider two main strategies when searching for ensembles: exhaustive and greedy. These search over model candidates with different hyperparameters as described in (Pan et al., 2019). Note that we also considered a "simple greedy" approach where the k best models on dev were selected, however this underperformed other approaches by 1 - 2 F1 points.

In exhaustive search we consider all possible ensembles, whereas in greedy search we build the ensemble one model at a time by looking for which model we can add to an i model ensemble to make the best $i+1$ model ensemble.

4.3.1 Exhaustive Search (ES)

In the exhaustive search approach where we consider each of the $\binom{m}{k}$ ensembles of k candidates from our group of m models. We then use our core strategy for each ensemble to obtain short and long answer F1 scores for each ensemble. After searching all possible ensembles we return two ensembles: (i) the ensemble with the highest long answer F1 score and (ii) the ensemble with the highest short answer F1 score.

4.3.2 Greedy Search (GS)

We select the models by greedily building $1, 2, ..., k$ model ensembles optimizing for short or long answer F1 using our core strategy.

4.4 Normalization Strategies

We investigate two primary methods for normalizing the scores predicted for a span: not normalizing and logistic regression. We also investigated normalizing by dividing the scores for a span by the sum of all scores for the span, however we omit these results for brevity as they did not produce interesting results.

4.4.1 None

As a baseline we run experiments where the scores for a span are used as-is.

4.4.2 Logistic Regression

We also experiment with normalization using logistic regression where the scores from the top prediction for the "dev-train" examples is used to predict whether the example is correctly answered. In our experiments using the top example performed equally well to using the top 20 predictions per example to train on. We also experimented with using other features which did not improve performance. To ensure an appropriate regularization strength is used, we use the scikit-learn (Pedregosa et al., 2011) implementation of logistic regression with stratified 5-fold cross-validation to select the L2 regularization strength.

4.5 Aggregation Strategies

We consider a number of aggregation strategies to produce a single span score for each span predicted by a system for an example. These include the baseline approach of max as well as the exponentially decaying sum, reciprocal rank sum, and noisy-or

methods influenced by IR. These approaches operate on a vector P of scores on which one of the above normalization strategies has been applied.

4.5.1 Max

For a vector P, $score = \max_{i=1}^{|P|} P_i$.

4.5.2 Exponential Sum (ExS)

Based on the ideas of (Young, 1974; Cormack et al., 2009), we sort P in descending order and take

$$score = \sum_{i=1}^{|P|} P_i * \beta^{i-1}$$

for some constant β (we use $\beta = 0.5$).

4.5.3 Reciprocal Rank Sum (RRS)

Based on the ideas of (Cormack et al., 2009), we sort P in descending order and take

$$score = \sum_{i=1}^{|P|} P_i * \frac{1}{i}$$

4.5.4 Noisy-Or (NO)

Based on the ideas of (Kurland and Culpepper, 2018; Wu, 2012), we take

$$score = 1 - \prod_{i=1}^{|P|} (1 - P_i)$$

5 Experiments and Results

We examine two types of ensembling experiments: (i) ensembling the same model trained with different seeds and (ii) ensembling different models. Ensembling the same model trained on different seeds attempts to smooth the variance to produce a stronger result. On the other hand ensembling different models attempts to find models that may not be the strongest individually but harmonize well to produce strong results. Throughout this section we will use $SA\ F1$ and $LA\ F1$ to denote the short and long answer performance on "dev-test". Similarly we will use N_S to indicate the number of models searched for an experiment and types SA and LA to indicate optimization for SA and LA F1 respectively.

5.1 Seed experiments

In Table 1 we find that there is a benefit to ensembling multiple versions of the same model trained with different seeds. Note that there is some data snooping ocuring here as the model is selected based on full dev performance (which is a superset of "dev-test"). RikiNet (Liu et al., 2020) and the 1 model performance reported above represent the top published NQ models at the time of writing this paper.

# Models	$SA\ F1$	$LA\ F1$
1	56.1	67.1
4	**58.7**	**69.6**

Table 1: Ensembling the same model trained with different seeds.

Search	N_S	Type	$SA\ F1$	$LA\ F1$
-	-	-	58.7	69.6
ES	20	LA	59.6	70.5
ES	20	SA	59.6	70.0
GS	41	LA	**59.7**	**70.8**
GS	41	SA	59.1	69.8

Table 2: Comparison of Search Strategies. All experiments run without normalization using max aggregation. The first row is 4 seed ensemble from Table 1.

5.2 Main experiments

We investigate the different search strategies in Table 2. We find that the greedy approach performs best overall, with the greedy ensemble optimized for LA performance performing the best on both short and long answer F1. Note that the numbers seen here, particularly when optimizing greedily for long answer performance are higher than those observed for ensembling the same model with multiple seeds. We hypothesize the superior generalization of greedy is due to exhaustive search "overfitting". For the remainder of this paper we will focus on greedy search optimized for long answer to keep the number of experiments presented to a manageable level.

We investigate the impact of the IR inspired normalization strategies in Table 3. The max experiment is as-before run without normalization to greedily optimize for long answer F1. The other experiments here are normalized with logistic regression, as our experiments showed that not normalizing decreased performance. We find that using max aggregation results in the best short answer F1 whereas using normalized noisy-or aggregation results in the best long answer F1. Based on these results, we run a final experiment using unnormal-

Aggregator	$SA\ F1$	$LA\ F1$
Max	**59.7**	70.8
Exponential Sum	58.3	70.4
Reciprocal Rank Sum	57.3	70.7
Noisy-Or	57.3	**71.5**

Table 3: Comparison of IR inspired aggregation strategies. All experiments run with a greedy search strategy optimized exclusively for long answer F1 with logistic regression normalization (except max which is not normalized).

ized max for short answers and logistic regression normalized noisy-or works for long answers. We find that this approach produces the strongest performance for both short and long answers with 59.3 $SA\ F1$ and 71.5 $LA\ F1$. Consequently we use unnormalized max ensembling of GAAMA answers (as GAAMA works on short answers) from 4 models in ARES. These numbers translate to a full dev performance of 59.3 short answer F1 and 71.1 long answer F1, which represents an improvement of 2.3 short answer F1 and 4.0 long answer F1 over our best single model.

6 Ensemble Candidate Contributions

When doing manual error analysis on the NQ dev set, we do observe patterns suggesting that each of the ensemble components do bring different strengths over the single best model. For example, the Wikipedia article for *Salary Cap* contains multiple sentences related to the query *"when did the nfl adopt a salary cap"*:

> The new Collective Bargaining Agreement (CBA) formulated in 2011 had an initial salary cap of $120 million...The cap was first introduced for the 1994 season and was initially $34.6 million. Both the cap and...

The later sentence contains the correct answer, *1994*, since the question is asking for when the salary cap was initially adopted. One of our models A correctly makes this prediction whereas another one of our models B predicts *2011* from the earlier sentence. There are also cases where the correct answer span appears in the middle or later part of a paragraph and, though our model B predict the spans correctly, they assign a lower score (relative to its optimal threshold) than the model A. The position bias, therefore, appears to hurt the performance of the system in certain situations where location of the answer span relative to the paragraph is not a useful signal of correctness.

Finally, in our manual review we do see that the ensemble of these models performs better in the expected ways: (1) boosting scores when multiple models agree on an answer span even though no one model is extremely confident (2) reducing confidence when there is disagreement among models.

7 Conclusion

We introduce a novel concept for a MRC service, ARES, which uses an ensemble of models to respond to requests. This provides for multiple advantages over the traditional single model paradigm: improved F1, the ability to control the performance vs runtime tradeoff for each individual request, and improved explaiability of results by showing both candidate answers in addition to the final ensembled answer. We outline several ensembling approaches for question answering models investigated for use in ARES and compare their performance on the NQ challenge. Our findings show that ensembling unique models outperforms ensembling the same model trained with different seeds and provide further analysis to show how ensembling diverse models improves performance. We also show that using unnormalized max aggregation for short answers and logistic regression normalized noisy-or aggregation for long answers yields an F1 improvement of 2 to 4 points over single model performance on the NQ challenge.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. *CoRR*, abs/1906.05416.

Rishav Chakravarti, Cezar Pendus, Andrzej Sakrajda, Anthony Ferritto, Lin Pan, Michael Glass, Vittorio Castelli, J William Murdock, Radu Florian, Salim Roukos, and et al. 2019. Cfo: A framework for building production nlp systems. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 758–759, New York, NY, USA. ACM.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of

deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Natural Language Computing Group. 2017. R-net: Machine reading comprehension with self-matching networks.

Oren Kurland and J. Culpepper. 2018. Fusion in information retrieval: Sigir 2018 half-day tutorial. pages 1383–1386.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: a benchmark for question answering research. *TACL*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. 2020. Rikinet: Reading wikipedia pages for natural question answering.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou, and Jianping Shen. 2019. An end-to-end dialogue state tracking system with machine reading comprehension and wide & deep classification.

Lin Pan, Rishav Chakravarti, Anthony Ferritto, Michael Glass, Alfio Gliozzo, Salim Roukos, Radu Florian, and Avirup Sil. 2019. Frustratingly easy natural question answering.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W. Bruce Croft, and Mohit Iyyer. 2020. Open-retrieval conversational question answering.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *EMNLP*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Varun Talvar. 2016. grpc design and implementation. Talk by Varun Talwar, Product Manager at Google at Stanford, California [Accessed: 2019 06 20].

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Shengli Wu. 2012. *Data Fusion in Information Retrieval*. Springer Publishing Company, Incorporated.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019a. End-to-end open-domain question answering with. *Proceedings of the 2019 Conference of the North*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019b. XLNet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

H.P Young. 1974. An axiomatization of borda's rule. *Journal of Economic Theory*, 9(1):43 – 52.