BMTURK: Crowdsourcing System on Blockchain

ChangSeok Oh, Linh Hoang, Kedi Zheng
Georgia Institute of Technology

Overview

Crowdsourcing is a successful sourcing model that allows people to cooperate a difficult task looking nearly impossible in perspective of its scale or required resources. Amazon Mechanical Turk proves the effectiveness of the crowdsourcing model. However, current crowdsourcing services and platforms are centralized so they need a central authority that controls the process between workers and requesters. This centralized platforms inevitably charge a service fee or a commission for maintenance. We tackle this cost problem with blockchain technology. In this project, we develop BMTURK, a simplified crowdsourcing system that runs on Ethereum network. To reveal its strength and weakness, we evaluated BMTURK in terms of latency and cost. The latency for creating a contract (a job request) was 40–70 seconds while varying the number of words of the contract from 500 to 1,000. And the cost for a contract of 1,000 words was $\$0.016$ in BMTURK with IPFS, which was $750\times$ cheaper than one of Amazon Mechanical Turk.

1 Introduction

Crowdsourcing is one of sourcing models that describe how individuals or organizations supply necessary resources such as idea, goods, finances, human labors, etc. In this model, a huge job that looks nearly impossible to be achieved by an individual is split into multiple small tasks, and many participants who are interested in the job on internet perform them independently [10]. This idea sounds clever but not brand-new. We have observed many applications of the crowdsourcing model already (e.g., Wikipedia, LEGO ideas, Airbnb, etc.).

In 2005, Amazon introduced a crowdsourcing platform called Amazon Mechanical Turk. The key idea of the service is to provide a centralized marketplace with a rewarding system [1]. It is a brilliant idea, and seems to achieve certain success. However, we think the centralized labor marketplace still has some issues. First, the centripetal service charges expensive fees for maintenance (up to 40% depending on the amount of work). Second, Amazon Mechanical Turk does not provide a way to verify the quality of completed jobs. What job requesters can do is to just rely on a consensus among works who did the same work. Third, a generic interface is not provided, so job requesters should spend extra money to hire software developers. Lastly, Amazon's labor trading system only accepts a traditional banking system that prevents many of workers who do not want to reveal themselves from participating the crowdsourcing.

We will tackle these problems, using blockchain technology. Theoretically speaking, the blockchain should not suffer from aforementioned problems since it follows a distributed transaction model. There is no central authority that controls requests and works in the distributed model so that neither commission nor maintenance fee is required. Also, we can combine any banking system with the internal currency of blockchain. Some researchers and companies already did similar jobs [4, 8]. However, they are either proprietary or close-sourced so we could not access their work to analyze the blockchain-based crowdsourcing system.

This project aims to prototype our own crowdsourcing system running on blockchain technology, and identify its pros and cons. Clearly speaking, our goal is not to build a perfect and secure crowdsourcing system on the blockchain network, rather to explore possibilities and limitations of the technology when implementing crowdsourcing system on top of blockchain technology.

We present the design of our blockchain based crowdsourcing system in §2, and describe implementation details in §3. In §4, we deal with evaluation results of our crowdsourcing system with blockchain technology. Lastly, we summarize our findings and achievement in §5.

2 Design

We prototyped a blockchain-based crowdsourcing system in BMTURK, using Ethereum platform. Figure 1 shows overall system composition and design overview of BMTURK. There are three distinct roles in our system: a requester, a worker and a verifier. But they are performed by all participants on the crowdsourcing overlay network, and implemented as a distributed application for Ethereum platform. It follows four steps: (1) A requester creates a job (i.e., a contract), and posts it in the network; (2) Workers find the contract and see which task is available; (3) Workers conduct part of works for the contract and submit them at will; (4) Verifiers pick up the works done by workers, validate them, and notify results to task performers. We classified this procedure, realized each step into adequate protocol, and implemented the protocol on top of Ethereum platform. In addition, to help participants expedite each step, we built user interface by using web technology.

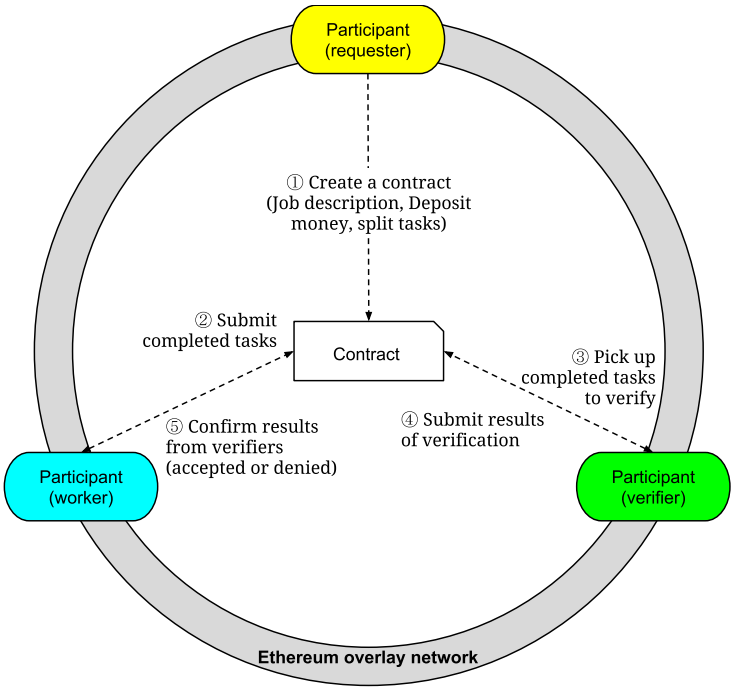

Figure 1: Design Overview of BMTURK

Figure 2 shows a simplified sequential workflow of BMTURK. The workflow starts with a participant who create a job to request. (i.e., a contract.) When creating a contract, the requester puts detailed information such as job description, success criteria, reward, the number of split tasks, and so on. The created contract

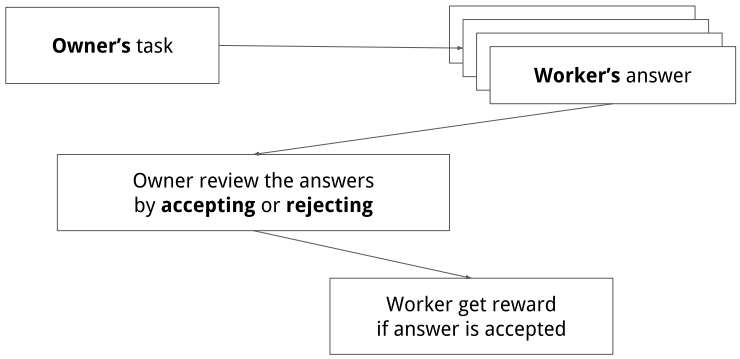

Figure 2: BMTURK workflow

is deployed to other participants in the network. When it is done, every participants can see what tasks are available in user interface we provide, conduct a task, and submit a completed task to the system. Then, the submitted tasks spread into the network again to get verifications from other participants (i.e., verifier). Once a completed task is inspected, the result is notified to the workers who completed the task. BMTURK displays the result as either 'accepted' or 'rejected'. For rejected tasks, a worker can choose either re-working or discarding the task. We designed that rewards are granted when all completed tasks are accepted, and the rewards are paid from the initial fund of the contract. Where the money runs out, the requester can put more money for the contract later. This is the basic workflow of BMTURK.

We admit this system might not handle all edge cases that can be exploited by a malicious participant. For example, he can intentionally repeat incorrect verification jobs to seek verification fees. To prevent this, BMTURK may need a kind of reputation system. However, this is out of scope. Our goal is to analyze pros and cons of a blockchain based crowdsourcing system, not build a perfect crowdsourcing system. We leave the reputation system as a future work.

3 Implementation

For fast proof-of-concept, we simplified the whole design, but implemented essential protocols to make BMTURK similar to Amazon Mechanical Turk. The simplified workflow of BMTURK also begins from a job requester who creates a contract for crowdsourcing. In the contract, the requester can leave a job description, a compensation (e.g., amount of money), and the necessary number of workers. Once the job requester committed a contract, it is deployed across the Ethereum network and posted on a public bulletin. Workers can choose a suitable task according to their skill set, interests, and the reward. When they submit answers after finishing their tasks, the job requester can decide whether or not he accepts the answers. If he does, the workers whose answers are accepted will receive the promised compensation. Otherwise, another or same worker can retry rejected jobs.

BMTURK consists of two parts: a back-end and a frontend. We implemented the back-end, using IPFS [5] on the Rinkeby network which is a test net for Ethereum blockchain. It uses **proof-of-authority consensus algorithm** [6]. The IPFS is a peer-to-peer distributed file system that aims to connect all computing devices into one file system. We used it to lower the cost of deploying information on the blockchain as each task description can be represented by a 256-bit SHA-1 hash. Because this hash can be used to download the file from IPFS nodes, it significantly lowers the cost of transferring data on the blockchain. To make the back-end work on the Rinkeby, we implemented two smart contracts: (1) JobSet, (2) Job. Figure 3 shows relation

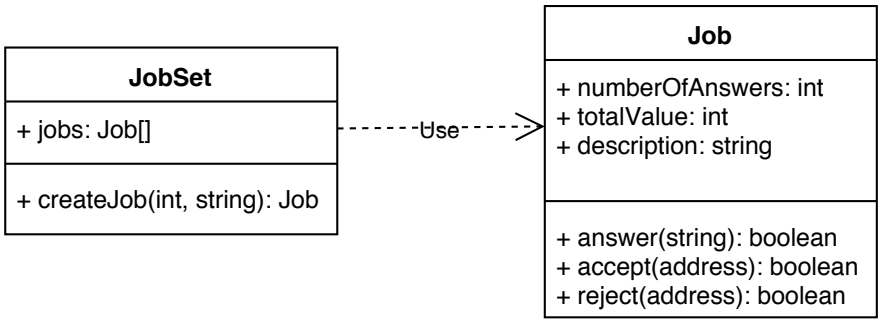

Figure 3: Smart contract design

between the two contracts. The JobSet represents a contract created by a requester so that it contains a full list of job instances. The two smart contracts provide basic but same functionalities of Amazon Mechanical Turk. We detailed each protocol in Table 1.

Protocol	Description
JobSet	
jobs: Job[]	The list of all crowdsourcing contracts
createJob(int, string): Job	The method to create a crowdsourcing contract
Job	
numberOfAnswers: int	The number of needed answers
totalValue: int	The total rewards of this contract
description: string	The description of this contract
answer(string): boolean	The method used by the worker to submit his work
accept(address): boolean	The method used by the owner to accept an answer
reject(address): boolean	The method used by the owner to reject an answer

Table 1: Details of BMTURK's smart contracts

On the other had, the frontend of BMTURK is implemented by using web technology (i.e., HTML, CSS, JavaScript). It can be either hosted on a web server or locally run on users' machine after download. Two browser extensions: MetaMask and IPFS are necessary to run BMTURK. MetaMask is a browser extension to connect to a blockchain node, and IPFS companion is used to connect with a IPFS node. BMTURK provides two views for users to facilitate crowdsourcing works. One is a browsing view that shows the full list of contracts, and the other is a contract view that shows details on each contract.

Figure 4 shows a snapshot of the browsing view. The browsing view page shows a full list of crowdsourcing contracts deployed on the Ethereum network on its top, and provides an user interface to create a crowdsourcing contract (i.e., a job request) below. To create the contract, a job requester needs to input a task description, the reward to pay for each accepted answer, and the number of answers he wants into the given text fields.

Figure 5 and Figure 6 show contract views for a job requester and a worker, respectively. Both show information about the selected contract (e.g., task description, reward, etc), and whether or not the contract is accepting answers. The job requester view shows the list of answers and states of each answer. A job requester can review submitted answers by clicking buttons corresponding to Accept and Reject. On the other hand, the worker view provides an user interface to newly submit answers or check if an answer was

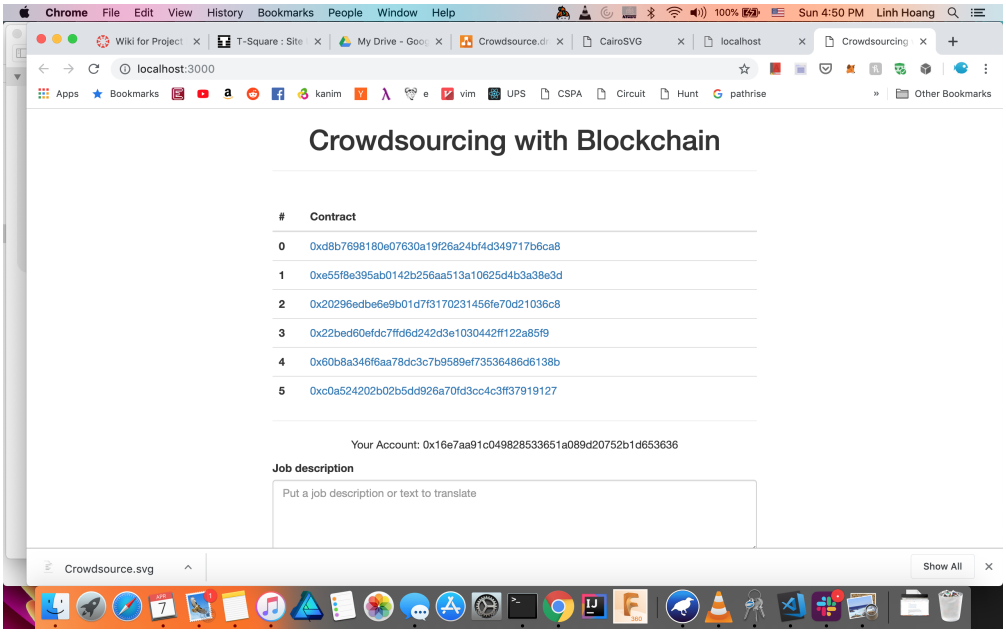

Figure 4: The browsing view

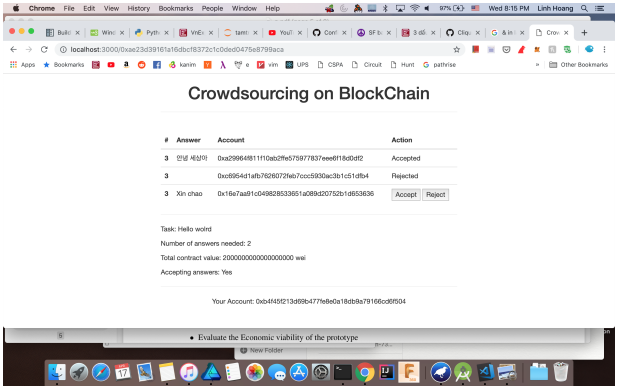

Figure 5: The job requester's contract view

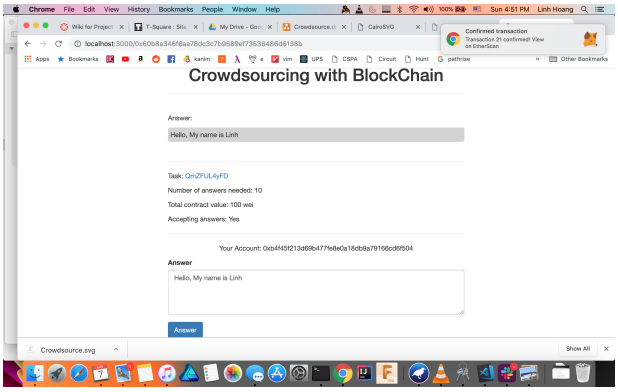

Figure 6: The worker's contract view

submitted before. In addition, it helps workers see whether their submitted answers are accepted or rejected..

4 Evaluation

We prototype a small Ethereum application that translates an article into multiple languages by using a crowdsourcing manner, and evaluated BMTURK with it in terms of latency and economic benefit. For simplicity, we focus on showing feasibility and functionality rather than making BMTURK perfect. So, other metrics such as security, performance, overhead are out of scope. We leave them as future works.

4.1 Experiment setup

We developed BMTURK on top of ganache, and performed experiments with two remote machines on the Rinkeby test net. They played a job requester and a worker respectively. We assume the performance of two

participants in BMTURK would be similar to one of more participants because they just serve as thin clients on the Ethereum network.

4.2 Translation in crowdsourcing

We decided to develop a distributed application for translation on the Ethereum network since it was the first example showing the effectiveness of crowdsourcing [11]. We also thought that translating would be a good example for BMTURK because of its characteristics like followings. First, translating work can be split up into multiple sub-tasks without loss of the original information. Second, it does not require too much technical knowledge. Third, the peculiarity and informality of human languages make the translation hard for machines for now. For these reasons, we thought translating a text in a crowdsourcing manner on blockchain network would be a good application for showing the effectiveness of BMTURK.

We tried to translate English articles of Wikipedia into Chinese for evaluating BMTURK, While varying the length of each article, we measured the cost and the latency.

4.3 Data set for evaluation

We excerpted paragraphs from a Wikipedia page [9], and carefully picked some of them that consist of 500-1,000 words. It is worth to note that plain text is used in this experiment instead of IPFS. This is because the size of plain text is much bigger than that of IPFS link so that we could measure how the blockchain network performs with such heavy payload. Another reason is that it was hard for us to keep connections stable in the university. that blocks P2P network that IPFS relies on.

4.4 Latency

To measure the latency, we calculated end-to-end time (i.e., from creating a contract creation to receiving the contract) between two remote machines, varying the length of payload for a contract.

Figure 7 shows the result of our latency experiment. There are two lines in the figure: a green line and red line. The green line represents local latency (i.e., how long it takes for a job requester to see his own request on the blockchain). The red line represents remote latency (i.e., how long it takes for workers to see the contract made by a job requester on the blockchain). As the payload of a contract increases, so does the latency. We observed that the latency ranges from 40–70 seconds.

We noticed the latency dipped at 700 words. This is because we used a public test network via MetaMask for evaluation. The public blockchain network is more prone to be affected by network load than the private blockchain network. Other than that, the green and red lines are closely coupled with each other since participants are connected on the same MetaMask network, and the network notifies them only when a transaction (i.e., payload of a contract) is completely propagated to all nodes in the network.

40–70 seconds for a single transaction is not a good news in performance perspectives. However, we blame plain text for this low latency. As mentioned before, the plain text causes kilobyte-sized data in payload. If IPFS was used, the payload would be reduced to 100 bytes at most.

4.5 Cost

BMTURK runs on the Ethereum network so BMTURK's users need to pay miners for verification works on their transactions. We can consider this cost, called **gas** in Ethereum terms, the real gas for automobiles. It propels payloads, i.e., the heavier a payload is, the more gas is required. The cost calculated in ETH (Ethereum) per transaction can vary depending on the payload of a contract.

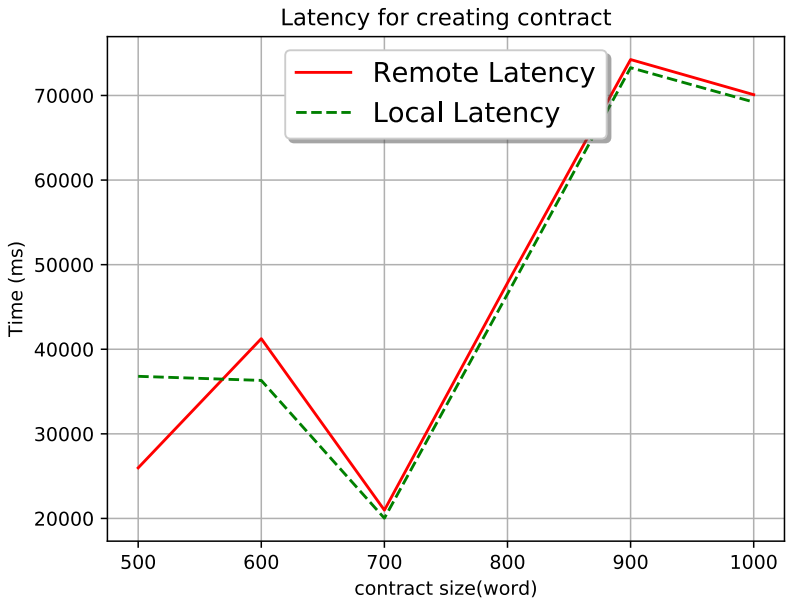

Figure 7: Latency for creating a contract while its word count varies from 500 to 1,000

In Figure 8, we can see the price did not move as much as the payload varies. The payloads for 500 and 1,000 words belong to the same magnitude so that we could not find many differences in the cost. If we switched to IPFS, the cost could drop down to 0.0001 ETH per transaction. At the time of this writing, 1 ETH coin is worth $166.62 [3]. That brings our transaction to $0.83 for plain text and $0.016 for IPFS.

4.6 Economic viability over other crowdsourcing services

There are some crowdsourcing services in the market, which provide a platform for clients to exchange their labors and resources in the market. The centralized vendors usually take a commission out of the contract. For example, Amazon's Mechanical Turk charges $20 - -40\%$ depending on the type of tasks [2]. The price for translating a sentence composed of 10 words is about $0.10 each [7]. If we translate 100 sentences, total $12 are charged (i.e., $10 for translation and additional $2 for commission). However, BMTURK with IPFS would cost just $0.016 for the same work.

Based on this finding, we concluded that the cost benefit of BMTURK becomes bigger as the task becomes bigger. BMTURK could keep the cost constant due to IPFS while the cost in Amazon's Mechanical Turk is raised as the size of translating jobs gets bigger and bigger.

5 Conclusion

Crowdsourcing is effective but costly. BMTURK, a crowdsourcing system on blockchain could address the expensive cost problem. We proposed a necessary set of protocol, and implemented them BMTURK. As BMTURK is compatible to any Ethereum network, it can play as a crowdsourcing platform. We proved its feasibility and effectiveness via a crowdsourcing application for translation. We evaluated BMTURK with the application in terms of latency and cost. The latency of BMTURK for transactions was 40–70 seconds,

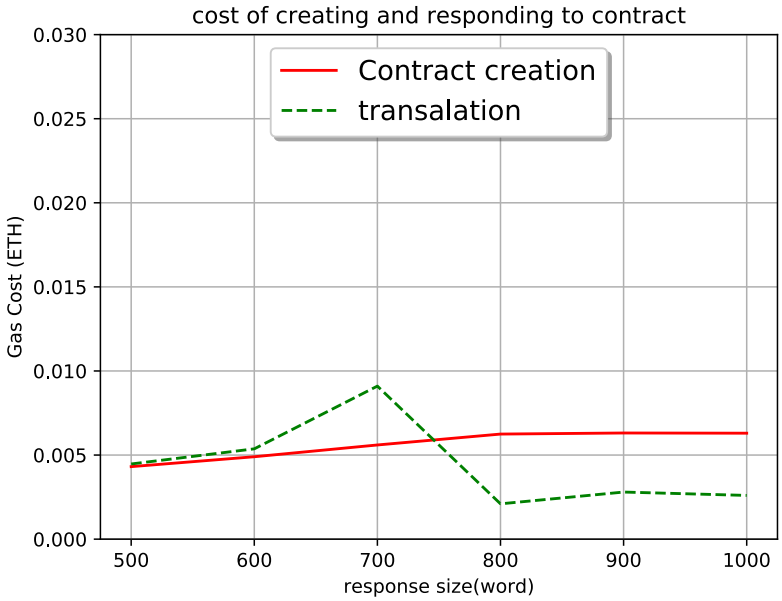

Figure 8: Cost in ETH for a contract that consists of 500–1,000 words

varying the number of words from 500 to 1,000. That sounds not ideal, but it is almost negligible if we consider the time for a single human to do the translation job alone. Also, we found that the strength of crowdsourcing system that runs on blockchain network is mostly aligned with the cost. It does not require any commission or a maintenance fee. Even more, it can keep the rate for deploying a job request to the entire network flat. The price for creating and deploying a contract that consists of 1,000 words is about $0.016 in BMTURK with IPFS. That is 750× cheaper than Amazon Mechanical Turk.

References

[1] Amazon. MTurk, 2019. https://www.mturk.com.

[2] Amazon. mechnicalTurk-pricing, 2019. https://www.mturk.com/pricing.

[3] C. Base. ETH-price, 2019. https://www.coinbase.com/price/ethereum.

[4] icrodrops.com. Gems, 2019. https://icodrops.com/gems.

[5] IPFS. IPFS, 2019. http://ipfs.io.

[6] karalabe. Clique PoA protocol & Rinkeby PoA testnet, 2019. https://github.com/ethereum/EIPs/issues/225.

[7] C. C.-B. Michael Bloodgood. Using Mechanical Turk to Build Machine Translation Evaluation Sets, 2012. http://cis.upenn.edu/~ccb/publications/using-mechanical-turk-to-build-machine-translation-evaluation-sets.pdf.

[8] Microwork. Microwork, 2019. http://www.microwork.io.

[9] Wikipedia. translated-article, 2019. https://en.wikipedia.org/wiki/Vietnam.

[10] Wikipedia. Crowdsourcing, 2019. https://en.wikipedia.org/wiki/Crowdsourcing.

[11] Wikipedia. wiki-translation, 2019. https://en.wikipedia.org/wiki/Wikipedia:Translate_us.