Investigation of Human-Computer Task Markets: Methods and Prototype

Dafna Shahaf[*] Eric Horvitz[†]

December 9, 2009

Abstract

We present research on task markets where automated planning procedures are used to enlist computational and human expertise to jointly contribute to the solution of problems, based on a consideration of the competencies, availabilities, and pricing of the different problem-solving resources. The methods meld the area of *human computation* with automated reasoning. We describe a prototype that creates plans for harnessing people and computation to perform translation among multiple languages. With the prototype, people with different skills are recruited to refine rough translations generated by a machine translation system. We present details about the hardness of plan generation, provide methods for solving them, and review experiences with the use of a prototype.

1 Introduction

There has been growing interest in an area of research referred to as *human computation* [vA08]. The work centers on the recruitment of people to perform tasks implicitly while they are engaged in online games. Human computation is also central in *task markets*, such as Amazon's Mechanical Turk (mTurk.com), where methods are provided online for specifying, recruiting, and reimbursing people to perform problem-solving tasks. We present an effort to generalize human computation to *human-computer computation*. We explore the feasibility of creating task markets that recruit both human and machine intelligence. We focus in this paper on the challenge of generating plans that assign subtasks to different agents, spanning human and computational problem solvers.

Human-computer computation (HCC) draws on efforts in *complementary computing* [HP07, Hor07], focused on the development of methods that explicitly consider the competencies and availabilities of both computational and human resources to solve problems at hand.

[*]Computer Science Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213. `dshahaf@cs.cmu.edu`. Research performed during an internship at Microsoft Research.

[†]Adaptive Systems & Interaction Group of Microsoft Research, 1 Microsoft Way, Redmond, WA 98052. `horvitz@microsoft.com`.

HCC task markets might one day operate within ecosystems that provide pools of candidate human and machine problem-solving agents and enable the dynamic creation of custom-tailored solvers. On the way to that potential future, several challenges must be addressed. Learning, reasoning, and optimization will play a critical role in making intelligence markets a reality . HCC challenges include the development of components that interpret posed problems and then identify, recruit, and reimburse sets of human and computational problem solvers to solve them. Effective HCC systems and platforms will require abilities to learn about competencies, availabilities, and pricing of the different problem-solving resources..

In this paper, we review the HCC opportunity and focus on the core challenge of HCC plan generation. HCC plans are aimed at "federating" the solution of subproblems via their distribution to multiple expert solvers, and then weaving together the answers into overall solutions. We focus on the complexity of and solutions to the plan generation problem for a class of HCC problems.

We shall describe key aspects of the challenge of developing problem-solving platform for the task of language translation. We focus on a prototype, named Lingua Mechanica, that creates plans for acquiring translation from people and computation to perform translation among multiple languages. In Lingua Mechanica, people with different translation competencies are recruited to refine rough translations generated by a machine translation system. The system continues to optimize its distribution of subproblems based on the expected utility of engaging human assistance, and continues to collect data to learn about competency and availability.

After an overview of key components of an HCC platform and service, we discuss the hardness of the HCC planning problem for language translation. We provide polynomial approximations to several important classes of models. Finally, we review how we can use games to harness human computation as means for incenting people to refine the output of machine translation.

2 HCC Planning Problem

An HCC platform includes components for accepting, interpreting, and decomposing a task into subproblems and for generating task federation plans (see Figure 1). An HCC planner accesses information about the availabilities, abilities, and costs (and other preference information) of agents that can provide problem-solving effort in return for some reimbursement that incents them to solve task subproblems. Execution is monitored and data is collected about multiple aspects of the problem solving. Learning from data can be harnessed to build predictive models about such attributes as the abilities, availabilities, and preferences of agents. We focus here on the planning problem for harnessing people and computation to perform translation among multiple languages.

Translation is a fruitful domain for human-machine collaboration: translating long documents is often a cumbersome task, requiring skilled bilingual people. Although machine translation (MT) has made significant progress in recent years, human aid is still essential for natural-sounding results (e.g. [CBBS04]). See Figure 2 for an example of an automated French-to-English translator: the translation is far from perfect, but provides understandable output to English speakers. Moreover, providing fixes for

MT output is easier, requiring less skill and can be performed by monolingual people.

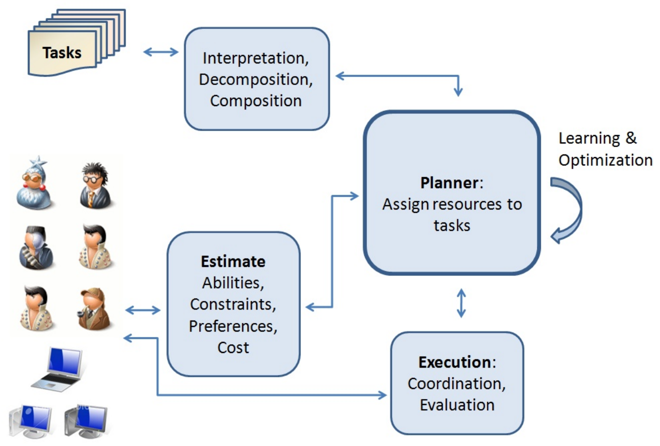

Tasks
Interpretation, Decomposition, Composition
Planner: Assign resources to tasks
Learning & Optimization
Estimate Abilities, Constraints, Preferences, Cost
Execution: Coordination, Evaluation

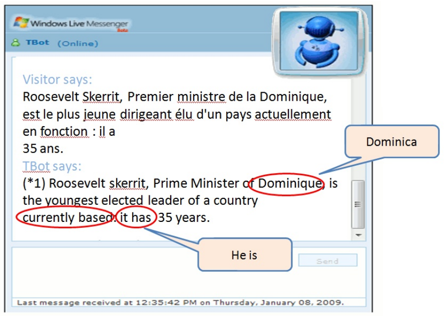
Figure 1: Problem-solving Platform

The task federation planning problem lies at the heart of HCC. We consider a set of n agents, $\mathbb{A} = \{A_1, ..., A_n\}$. Each agent A_i is associated with a set of abilities, $\{ability_j\}$. An ability is a pair $\langle b_k, v_k \rangle$ for b_k ability identifier and $v_k \in \mathbb{R}^+$ represents the skill level of the agent for that ability. Agents can represent any source of problem solving, spanning human and computational resources. Abilities can span a great spectrum of competency from such computational tasks as "Factor number N" to "Write a review for movie M."

Typical abilities for computational agents we employ at our translation CTM platform are of the form $\langle$"Machine Translation French$\gg$English,"$0.7\rangle$. We also have access to multilingual people with such bilingual skills as $\langle$"Human Translation French$\gg$English,"$0.9\rangle$ and monolingual people ($\langle$"Hindi speaker,"$0.95\rangle$).

In addition to agents, there is a set of m independent high-level tasks $H = \{h_1, ..., h_m\}$ and a set T of low-level tasks. Each high-level task h_i can be carried out in one or more ways (*scenarios*). A scenario is a directed acyclic graph (DAG), where vertices are low-level tasks, and edges specify a partial precedence order: there is an edge from t_i to t_j if task t_j cannot be performed unless task t_i is already satisfied (e.g. because its output is needed as an input to t_j). A low-level task t_j is associated with a set $demand_j$ of abilities.

Figure 3 displays an example of a high-level task. We wish to translate a paragraph from English to French. The figure shows three different scenarios: (a) acquire a bilingual person who can translate the paragraph directly, (b) acquire two bilingual people who can translate from English to French via an intermediate language (in this case, Italian), and (c) use machine translation procedures to translate the paragraph and then find a French-speaking person to correct the paragraph. Scenario B might be assumed a priori to require more competent language skills from both of the human translators, as the pipeline with two translations is likely to create more errors than single-stage translations.


Windows Live Messenger

TBot (Online)

Visitor says:
Roosevelt Skerrit, Premier ministre de la Dominique, est le plus jeune dirigeant élu d'un pays actuellement en fonction : il a
35 ans.

TBot says:
(*1) Roosevelt skerrit, Prime Minister of Dominique, is the youngest elected leader of a country currently based: it has 35 years.

Last message received at 12:35:42 PM on Thursday, January 08, 2009.

Figure 2: Machine Translation log. Errors are marked in circles, partial fixes in text balloons.

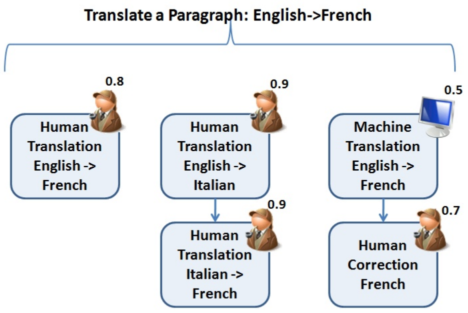

Translate a Paragraph: English->French
0.8
Human Translation English -> French
0.9
Human Translation English -> Italian
0.9
Human Translation Italian -> French
0.5
Machine Translation English -> French
0.7
Human Correction French

Figure 3: Three scenarios for translating: human translation, human translation through an intermediate language, machine translation fixed by a person. Numbers refer to desired quality.

A *coalition* $C \subset \mathbb{A}$ can be defined as a group of agents who have decided to cooperate in order to achieve a common task. We assume that a coalition can work on a single (low-level) task at a time. We consider both cases in which agents may or may not be members of more than one coalition.

The utility gained from performing the task t_j by coalition C is $U(C, t_j)$. The utility depends on tasks demands and the coalition's combined abilities. The utility of a high-level task h_i depends on the utilities of the low-level tasks it is composed of.

One common framework assumes modularity — the combined ability set of a coalition is defined as the sum of abilities of participating agents. If agents participate in more than one coalition, they can distribute their abilities between the coalitions. In this framework, t_j can be satisfied by a coalition of agents C if their combined abilities are higher than the task's demands, and $U(C, t_j) = 0$ otherwise. A high-level task h_i is satisfied if all of the low-level tasks t_j of at least one scenario have to be satisfied. Later in this paper we consider other types of utility functions.

Definition 2.1 (Coalition Problem)**.** *Given* $< \mathbb{A}, H, T >$, *the Coalition problem is to assign tasks* $t \in T$ *to coalitions of agents* $C \in \mathbb{A}$ *such that the total utility is maximaized and the precedence order is respected.*

In general, agents and tasks are both associated with other sets of constraints (e.g.

money and availability). In this more-general case, the problem is to maximize the utility under all the constraints.

3 Hardness and Approximations for the Planning Problem

In this section, we discuss the hardness of the HCC planning problem and provide polynomial approximations under various models. We conclude the section with a discussion on applying these algorithms in practice to our prototype system.

3.1 Simple Tasks

We start with the simplest type of tasks: a high-level task h_i is composed of nothing but a single scenario, which is composed of a single low-level task t_i.

3.1.1 Hardness

The problem is $\mathcal{NP}$-hard. The decision problem is in $\mathcal{NP}$ (given a solution, it can be verified in polytime). We show a reduction from weighted exact set cover.

Definition 3.1. *Given a set X and a set $\mathcal{S}$ of subsets of X with associated rewards $R(S)$ for each $S \in \mathcal{S}$, an exact set cover $\mathcal{S}^*$ is a subset of $\mathcal{S}$ such that every element in X is contained in exactly one set in $\mathcal{S}^*$. The goal is to find the exact cover with the maximal reward.*

Weighted exact set-cover is NP-hard. Given a weighted exact set-cover instance, we construct a Coalition instance:
Let $\mathbb{A} = X$. Each $S_j \in \mathcal{S}$ defines a high-level task h_j. h_j is composed of a single scenario and a single low-level task t_j, requiring $\langle b_j, |S_j| \rangle$ ($|S_j|$ units of ability b_j). Each $A_i \in S_j$ has 1 unit of ability b_j ($\langle b_j, 1 \rangle$). Therefore, the only coalition able to perform t_j (and thus, h_j) is S_j. Define the evaluation function such that $\sum U(S_j, t_j) = R(S_j)$.

A solution to the Coalition problem corresponds exactly to a solution to weighted exact set cover. Therefore, the problem is $\mathcal{NP}$-hard.

3.1.2 Tractable Approximations: Limiting Coalition Size

One of the factors contributing to the hardness of the HCC planning problem for translation is that the number of possible coalitions is exponential in n. Thus, a natural way to reduce the search space is to restrict the maximal size of a coalition to k, thus reducing the number of coalitions to $O(n^k)$ (but the problem is still $\mathcal{NP}$-hard). This assumption is very reasonable in our system – most tasks we deal with do not require more than a few participants.

We can represent this as a graph, in which the agents are vertices and tasks are a collection of weighted hyperedges. A hyperedge corresponds to assigning these agents to the task; its weight is the expected reward. This is also a useful framework for the case of extra constrains (e.g., agent availability): in this case, hyperedges correspond to the possible coalitions.

In addition, every task is identified with a color, and all of its edges are of this color. We want to find a maximum value matching in the graph s.t. only one edge of each color is used.

Claim 3.2. *The greedy algorithm is a constant-factor approximation to the Coalition problem (k is constant).*

The proof is based on standard greedy analysis. Any coalition chosen for our solution contains at most k agents, and we compare their assignment to the optimal solution's assignment.

3.1.3 Tractable Approximations: Special Utility Functions

We now withdraw the assumption of limited coalition size. The number of coalitions we need to consider is exponential in n again. Therefore, we have to make some assumptions on the utility functions to restrict our search space. Some common assumptions are submodularity (diminishing returns, $u_i(S \cap S') + u_i(S \cup S') \leq u_i(S) + u_i(S')$), subadditivity ($u_i(S \cup S') \leq u_i(S) + u_i(S')$) or superadditivity.

For example, suppose the utility functions are monotone and submodular.

Claim 3.3. *A greedy algorithm yields a $\frac{1}{2}$-approximation. In special cases where the the submodular function is of a special type (discussed later), a $(1 - \frac{1}{e})$- approximation has been achieved; this is optimal for these problems.*

Those results follow from efficient approximation algorithms [NWF78, Von08] that exist for the Submodular Welfare Problem. The Coalition problem can be easily formalized as a Welfare Maximization Problem. In this problem, m items are to be distributed among n players with utility functions $u_i : 2^{[m]} \rightarrow R^+$. Assuming that player i receives a set of items S_i, we wish to maximize the total utility $\sum_i u_i(S_i)$. In our case, an agent corresponds to an item, tasks correspond to players. For task t_i, utility function $u_i(C)$ is $U(C, t_i)$.

Similarly, the case of subadditive functions was handled by [Fei06], who relaxes the problem into an LP, and proposes a way of rounding any fractional solution achieving an approximation ratio of $\frac{1}{2}$. For the superadditive case an approximation ratio of $\frac{\sqrt{\log m}}{m}$ can be achieved.

3.2 Tasks Consisting of a Single Scenario

We shall now move to the more complex case where a high-level task h_i takes the form of a single scenario. The scenario DAG corresponds to a partial ordering on low-level tasks t_j.

Hardness of the problem follows from the previous section. We build on the algorithms described by Shehory and Kraus, to provide a constant-factor approximation to the case of limited-size coalitions. The algorithm can be viewed as a centralized version of the methods described in [SK95].

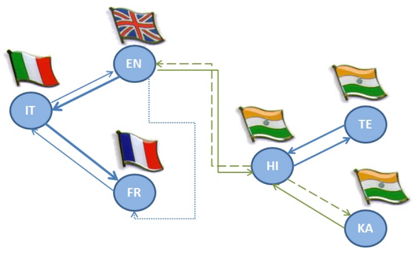


Figure 4: Translation abilities graph. Each resource is a collection of edges (in the same color). Dashed edges correspond to low skills.

For each task t, we denote its set of predecessors, including t, by $P(t)$. The choice of t for coalition formation will depend on the costs of and the benefits from, the formation of coalitions that perform all of the tasks in $P(t)$.

At each iteration, we choose in a greedy manner the task t with the maximal $u(P(t))$ to be performed together with all of its predecessors, and form the required coalitions. We remove t, all of its predecessors, and the assigned agents from the list, and iterate. We note that only a small portion of the calculations needs to be updated.

3.3 Multiple Scenarios

Let us now consider the case of multiple scenarios. As before, we assume the size of coalitions is limited. We also limit the maximal number of low-level tasks a scenario can involve.

The algorithm we propose is a combination of the two algorithms we have discussed. As before, we first construct a hypergraph. A high-level task is a set of hyperedges, corresponding to coalitions that can carry out one of the scenarios. Similar arguments hold for this case.

3.3.1 Case of Transitive Tasks

An interesting case of multiple scenarios for HCC solvers is where the tasks are *transitive*. Transitive tasks allow for a compact representation of many possible scenarios by encoding them as graph paths. Such transitivity is especially valuable in HCC for translation. The task of translating from French to English is the same as translating from French to any intermediate language, and then to English. The path might even involve a chain of translations among several intermediate languages.

A natural way to solve the case of transitive tasks is with *multi-commodity flow*. We have a weighted multigraph $G = (V, E, w)$. Each $s \in V$ represents a language. Each problem-solving resource, whether a human or computer-based reasoned, corresponds to a set of directed edges. An edge weight represents the user's ability to translate between the two. For example, in Figure 4 we have one user speaking Hindi and Telugu well, a Hindi speaker that knows some English and Kannada (dashed edges correspond to low weights on competency), and several European-language speakers.

In addition, we have k commodities (tasks), where each one has source language $s_i \in V$, sink (target language) $t_i \in V$, and a starting flow $d_i \in R$ (in the basic scenario,

$d_i = 1$). The goal of an HCC planners to maximize the total throughput.

There are two candidate paths for English-to-French translation (Figure 4): translation directly, or translating to French through Italian. Note that, given the solvers available, the direct edge indicates a low competency, but the other two edges are strong. However, as errors accrue over chains, the use of intermediate translations would tend to reduce the quality. How should we compare the output qualities?

In the HCC model for translation, the edge weights specifically represent the probability that the translation will reach a specified threshold of quality. Thus, the quality of a path is the *product* of its edge probabilities. We need to modify the multi-commodity flow solution to handle this case. Each directed edge e has f_i^{in} , f_i^{out}, and a capacity c_e. The key concept is that when flow goes through an edge, it loses some quality. We try to generate high quality at the sink. We first formalize this optimization problem as a linear program (LP), and then introduce rounding techniques to achieve a feasible solution:

$$\max \sum_i \sum_{e:\, \textit{edges to } t_i} f_i^{out}(e) \;\; \textit{s.t.}$$

$$\sum_i f_i^{in}(e) \leq c_e$$

$$\sum_{e:\, \textit{edges from } s_i} f_i^{in}(e) = d_i$$

$$f_i^{out}(e) = w(e) \cdot f_i^{in}(e) \quad \textit{(*Quality reduction*)}$$

$$\sum_w f_i^{out}(w,u) = \sum_v f_i^{in}(u,v) \quad u \neq s_i, t_i \quad \textit{(*Flow*)}$$

The HCC plan seeks to maximize the quality of commodity i reaching its sink. The flow on every edge (workload) is no more than the capacity of every edge. Alternatively, we can restrict total capacity per contributor.

Another interesting feature of transitive tasks is the possible re-use of intermediate results. Let us consider the case where we seek to have a Wikipedia article translated from English into both Kannada and Hindi. The previous algorithm would treat them as two separate problems, translating directly from English into both. However, given the resources and competencies available, it may be more beneficial to translate English to Hindi, and then Hindi to Kannada. Alternatively, it may be better to use an intermediate language and so travel via an intermediate node for both.

A similar flow formulation can be applied for re-using intermediate results. We define a variable for every $\langle$ language,task $\rangle$ combination (the quality of translation for a specific task in that language). A task would have one source and possibly many sinks. The goal now is to minimize the number of intermediate translations, while enforcing minimal translation quality in all sinks. We note that this minimization is a constraint, not the objective in this case. The objective should be interpreted as a routing constraint. Longer paths cost more and are prone to the accrual of errors. While quality is acceptable, we prefer shorter paths.

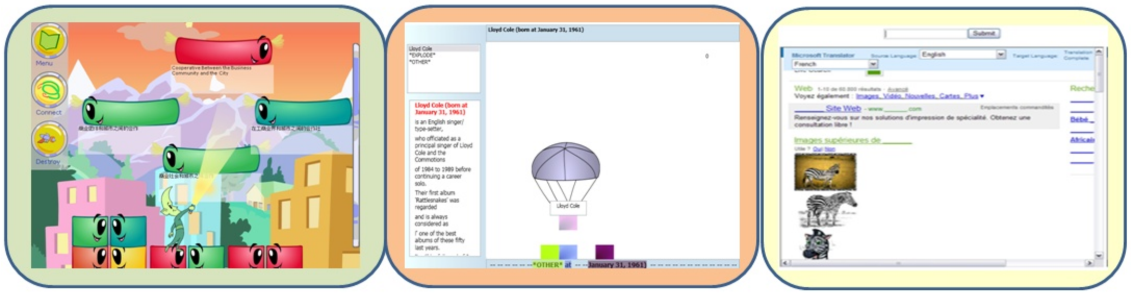


Figure 5: Games, left to right: Moon Climb, Word-Tetris, Dictionary Builder

3.4 Real-World Deployment

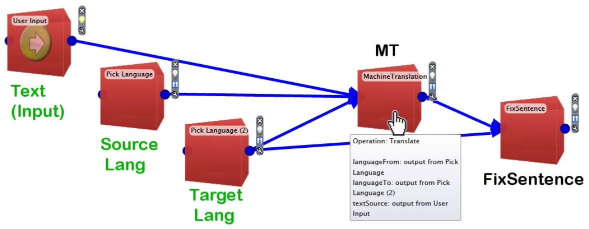
In the process of applying the above algorithms to our translation platform, several changes were made. First, in order to speed up the assignment process, we defined equivalence classes of abilities. Instead of using the fine-grained $\langle b_i, v_i \rangle$ abilities, we map them to equivalence classes (thus mapping agents and tasks). For example, we defined three levels of English proficiency. We ran the algorithms on a set of *representatives* for tasks and agents. When the greedy algorithm picked a coalition to perform a task, we would look for equivalent tasks and coalitions and assign as many as possible. The approximation guarantee still holds, and computation was significantly faster.

Our algorithms need to operate in an online setting, where users can log in and out at any minute. This has many implications, among them the need to employ dummy players (usually previous games recorded). When a task is assigned to a coalition, the dummy players are transparently replaced. Similarly, if one of the players disappears, another player (whether real or a dummy) immediately takes his place.

The online nature of the system makes learning availability patterns an attractive concept. The planner might not try to execute a translation to Chinese now if it knows some highly-skilled users will log in soon (e.g., since it is 4am now in China). We considered both learning those patterns and creating mechanisms that will let users report their patterns (especially for those who are getting paid for their work).

4 Implementation

We have implemented a system named Lingua Mechanica as a prototype HCC platform for performing translation among multiple languages. In this section we review the system. The system allows beginning-to-end creation of new tasks (with corresponding hierarchical scenarios) using a user-friendly mash-creator interface. The current implementation includes three games for engaging people to provide human computation. The games were developed with assistance from researchers in natural language processing (NLP). Our experiments with a pool of test subjects found that the system finds reasonable task assignments, and that the games often result in useful translations, created by an amalgam of machines and people.



Operation: Translate

languageFrom: output from Pick Language
languageTo: output from Pick Language (2)
textSource: output from User Input

Figure 6: Popfly-like interface for creating a scenario. The leftmost three blocks correspond to inputs. Others are functional blocks.

4.1 Requester Point of View

Let us assume that we wish to create a new task for Lingua Mechanica. The system currently supports two main methods, template-based and customized.

In order to facilitate task creation, the most common tasks are included in the system as templates. As an example, Lingua Mechanica provides a "Translate a page" option. The system engages the author with providing several task parameters, including the content that needs to be translated and the source and destination languages. In addition, the author of the task can change the metadata associated with the task, including parameters describing the desired quality, deadlines, and how much (if any) they are willing to pay for human contributions. Note that the cost and deadline affects the result: the planner can route the request through a professional translator (high-quality, slow, expensive) or through MT (low-quality, quick, cheap). The system can notify the user if the deadline seems unrealistic, based on history of similar tasks based on its growing database of experiences with translation.

Users can also define their own custom-tailored high-level tasks. Defining a new task involves the same data and metadata parameters, but also specifying the desired execution scenarios. We use a Popfly-like interface. Popfly.com is a website allowing users to create mashups. See Figure 6 for an example. The interface has pre-constructed blocks, which can represent any function our system supports. Blocks usually have parameters (e.g., arithmetic operation, or pair of languages) and one or more inputs. A mashup is created by connecting blocks together into a DAG, s.t. output of one block can be used in inputs of others. Any input should be mapped to some other block's output. This corresponds to our notion of *scenario* from Section 2. The mashup in Figure 6 corresponding to the rightmost scenario in Figure 3: the output of the "Machine Translation" block is the input of "Improve Sentence." If the user defines several alternative scenarios, the planner might be able to come up with a better solution.

4.2 Contributor Point of View

Users login to the system, and they can see their statistics and choose a type of a job. The types of jobs offered to them depend on their abilities. Users' abilities are determined through a short questionnaire (mostly used to determine the subset of languages they know) and through their performance: every now and then users are given a task for which we know the correct answer; this is used as a monitoring mechanism. In addition, while we are unsure about a user's abilities, we assign the same task to several others, until we achieve confidence. As an extra reality-check, we use a system developed by [CQ08], which scores sentences based on their correctness. Users who want to speed up the process (to get to more interesting, and perhaps better-paying tasks) can take *qualification* tests.

4.3 Games

Humans require some incentive to contribute to problem solving. Incentives include money, points, skill levels, community reputation, rankings, and even patriotism. Another emerging mechanism is online gaming: games are a seductive method for engaging people to participate in the computation process. This idea was first proposed by [vAD04], who used a matching game to get humans to label images.

After consulting with NLP researches, we have designed and implemented three simple games whose output may be useful to the NLP community displatey in Figure 5.

Word-Tetris: The goal of the game is fix a machine-translated paragraph. Two users (who speak the target language) are matched. Both see a sentence, broken into chunks attached to blocks falling from the sky. Each block may have several alternatives (corresponding to MT ambiguity: "il" is either "he" or "it," displayed in Figure 2). The initial choice is random per user. Users choose the relative location of the block (effectively choosing word ordering), and its phrase, applying wildcards/blanks when needed. Their goal is to match the sentence with their partner: a complete match would make a whole row disappear, while partial matches result in partial disappearance.

This was our most well-liked game. Users fixed MT-translated paragraphs from the French Wikipedia, matched with a dummy playing correctly. Most players became comfortable with the game within 1-3 rounds. Their translations were good at first, but declined as the game went faster and thus became more challenging. This is to be expected, and is compensated for by the verification mechanism of 2-player games.

Dictionary Builder: This game is meant to find a translation for words that do not appear in our bilingual dictionary, or words on which our MT systems have very low confidence. Given such a word/phrase, we search the web for images/snippets of text containing it (e.g., in Wikipedia or a monolingual dictionary). We MT this text, hiding the target word. The result is a collection of snippets in the target language with a missing word. We show them to two or more users; their goal is to match on the hidden word. For example, the word "*équipe*" (team) results in the snippet "A □ is a d' group; individuals partners with a common aim" and several images of football teams. A nice feature of this game is that it can seed itself. Suppose a word translation is discovered; if we have two monolingual dictionaries, we can align the corresponding definitions and find new translations.

Players grasped the concept of the game very quickly, and managed to match each other on a large fraction on the words (especially when searching for nouns; this might be attributed to the image search). We note that the quality of the snippets should be improved; when searching for words like '*étoile*" (star), the first snippets refer to "Etoile Restaurant," "Etoile Boutique," and several other stores – none of which provides a clue about the meaning of the word.

Moon Climb: Moon Climb is a simple matching game in which the goal is to reach the moon. You do so by matching flying creatures carrying similar sentences, causing them to fall down and form a tower. Moon Climb is a game for bilingual people, and is mostly used to asses their skill level (as a qualification test). Some users commented that it could also be suitable for children.

5 Conclusions

We have described research on developing a problem-solving platform that enlists both computational and human expertise to solve tasks. The system considers the competencies, availabilities, and pricing of the different solvers. We reviewed the challenge of harnessing people and computation to perform translation among multiple languages, focusing on the challenge of the ongoing generation and updating of plans that enlist people and machines to contribute. We adapted prior work to the challenge and showed how we can use a multi-commodity flow procedure in the plan generaetion. Finally, we reviewed the working Lingua Mechanica system and provided highlights in its use in authoring translation platforms and in recruiting human computation via engaging games. We foresee multiple challenges and opportunities on the horizon with continuing investigations of problem solving via human-computer computation and the creation and use of problem-solving platforms that can ideally tap human and machine intelligence.

Acknowledgements The authors would like to thank Vikram Dendi, Paul Koch, Raman Sarin, Paul Newson and Tommy A. Brosman for their assistance in developing the system.

References

[CBBS04] C. Callison-Burch, C. Bannard, and J. Schroeder. Improved statistical translation through editing. In *EAMT-2004*, 2004.

[CQ08] Colin Cherry and Chris Quirk. Discriminative, syntactic language modeling through latent svms. In *AMTA '08*, 2008.

[Fei06] U. Feige. On maximizing welfare when utility functions are subadditive. In *STOC '06*, 2006.

[Hor07] E. Horvitz. Reflections on challenges and promises of mixed-initiative interaction. *AI Magazine*, 28(2), 2007.

[HP07] E. Horvitz and T. Paek. Complementary computing: Policies for transferring callers from dialog systems to human receptionists. *User Modeling and User Adapted Interaction*, 2007.

[NWF78] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14(1), 1978.

[SK95] O. Shehory and S. Kraus. Task allocation via coalition formation among autonomous agents. In *IJCAI '95*, 1995.

[vA08] L. von Ahn. Human computation. In *ICDE*. IEEE, 2008.

[vAD04] L. von Ahn and L. Dabbish. Labeling images with a computer game. In *CHI*, 2004.

[Von08] J. Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC '08*, 2008.