Combining Information Extraction and Human Computing for Crowdsourced Knowledge Acquisition

Sarath Kumar Kondreddi
Database & Information Systems Group
Max Planck Institute for Informatics
Saarbrücken, Germany
Email: skondred@mpi-inf.mpg.de

Peter Triantafillou
School of Computing Science
University of Glasgow
Glasgow, Scotland
Email: peter.triantafillou@glasgow.ac.uk

Gerhard Weikum
Database & Information Systems Group
Max Planck Institute for Informatics
Saarbrücken, Germany
Email: weikum@mpi-inf.mpg.de

***Abstract*—Automatic information extraction (IE) enables the construction of very large knowledge bases (KBs), with relational facts on millions of entities from text corpora and Web sources. However, such KBs contain errors and they are far from being complete. This motivates the need for exploiting human intelligence and knowledge using crowd-based human computing (HC) for assessing the validity of facts and for gathering additional knowledge. This paper presents a novel system architecture, called Higgins, which shows how to effectively integrate an IE engine and a HC engine. Higgins generates game questions where players choose or fill in missing relations for subject-relation-object triples. For generating multiple-choice answer candidates, we have constructed a large dictionary of entity names and relational phrases, and have developed specifically designed statistical language models for phrase relatedness. To this end, we combine semantic resources like WordNet, ConceptNet, and others with statistics derived from a large Web corpus. We demonstrate the effectiveness of Higgins for knowledge acquisition by crowdsourced gathering of relationships between characters in narrative descriptions of movies and books.**

I. Introduction

A. Motivation and Challenges

Automatic *knowledge acquisition (KA)* from Web data and text sources has enabled the construction of huge knowledge bases (KB's) like DBpedia, Freebase, KnowItAll, NELL, Probase, YAGO, and more [3], [7], [10], [18], [44], [51]. When tapping into natural-language text, KA critically relies on *information extraction (IE)* technology, combining methods from pattern matching, computational linguistics, and statistical learning. *Open IE* methods [4], [8], [19], [24], [46] can derive a wide diversity of relational facts (instances of binary relationships) based on detecting and analyzing noun phrases (for entities) and verb-centric phrases (for relations), such as: "Saruman" "imprisons" "Gandalf". However, the same method also yields noisy or nonsensical triples such as: "Boromir" "takes" "Frodo" (from the sentence "Boromir tries to take the ring from Frodo."). Fully automated IE generally faces fundamental obstacles as input sentences often have complex structure, use of pronouns and other anaphoras, and ambiguous wording. Crowdsourcing, aka. *human computing (HC)* [28], is a natural alternative to overcome the fundamental limits of automated IE. It can tap human intelligence and knowledge, to assess candidate facts, to correct errors, or to add new facts.

HC has been applied to various tasks along these lines, including quality assessment of rankings, summaries, and IE results [1], [17], [25], query answering [22], [36], [37], entity matching [16], [50], ontology alignment [39], image search and annotation [15], [47], [52], and acquiring commonsense properties [42], [49] and taxonomies [26]. A typical approach is to place so-called *HITs* (human intelligence tasks), specifically designed micro-tasks, on platforms like Amazon's mturk [2] or CrowdFlower [13]. For certain problems, HC has been successfully leveraged in game form [48]: correcting OCR errors, annotating images, translating words to foreign languages, and so on. Notwithstanding all this prior work, we are not aware of any attempts so far to use HC and crowdsourcing platforms for the difficult KA task of extracting relationships between individual entities.

Employing HC for KA comes with opportunities and challenges. There is a great potential to acquire relational facts that are beyond hope for automatic IE. Humans watching a movie or reading a book can infer relations that may not explicitly appear in any specific sentence. However, humans alone cannot carry the burden of large-scale KA. The number of experts is limited, and these are not so likely to participate in online games. Hence, HC output will widely vary from high-quality to noisy and incorrect facts. Even if these HC errors could be compensated by redundant HITs and statistic reasoning over many contributors, there is still the issue of the total cost of such an approach: each HIT is a few cents only, but paying for hundred thousands or millions of HITs is very expensive if not prohibitive.

To overcome the limitations of either purely HC-based or purely IE-based KA, this paper puts forward a novel method that *combines* HC inputs with machine-centric IE. The key idea is to use automatic IE to generate questions and candidate answers (for multiple-choice questions), for a KA game. Our expectation is that this improves the quality of user contributions and lowers the cost of crowdsourcing.

B. Approach and Contribution

Our KA strategy for entity-relationship facts uses two stages.

1) IE phase: First we employ automated IE on appropriate Web corpora, in order to derive candidates for relation instances,

with an open set of potential relations. The output is a set of candidate triples for entity-relationship facts. We use a suite of techniques from computational linguistics, including dependency parsing (with the Stanford Parser) and pronoun resolution (with our own customized method). For high recall, our IE engine captures as many relational patterns as possible, and we also consider pattern-pattern relatedness by specifically designed *statistical (translation) language models (LM's)*. The resulting triples are usually of very mixed quality, necessitating the second stage.

2) HC phase: The noisy and incomplete sets of candidates from the IE stage and their underlying patterns are used to generate HITs in game form. Abstractly, each HIT presents the user with a *knowledge quad* of the form $(c, e1, r, e2)$ where $e1$ and $e2$ are entities, r is a relation, and c is a cue or textual context. One or more of the components $e1$, r, and $e2$ can be empty slots (variables) to be filled by the user; we may present a multiple-choice list to the user to pick the missing value. The quads are presented to the user in the form of questions, with candidate answers and additional free-text fields for entering further values. In this paper, we focus on the case where the relationship r is left to be filled, and both entities and the context are given. Our experiments specifically address relationships between characters in movies and books (with the IE stage based on narrative summaries).

A major challenge that our approach addresses is *sparseness of data*: relevant patterns for advanced relations tend to be rare. For example, phrases like "imprison", "orders to assassinate", or "pretends to fall in love" are infrequent even in a large corpus of movie plots and book summaries. We address this issue by combining *semantic resources* for relational phrases, specifically WordNet, ConceptNet, ReVerb, PATTY [19], [21], [32], [41], with *statistical translation models* that consider related phrases for better coverage.

Another challenge is *human weaknesses*: the quality of the HC contributors is bound to have high variance. To counter this problem, we use statistics to generate interesting questions about *important* entities and *salient but not obvious* relationships. Candidate answers for multiple-choice input are judiciously *ranked*, and an additional diversification step serves to avoid boring the user with near-duplicate choices.

The paper makes the following main contributions:

- a *principled framework* to combine IE and HC for KA tasks like compiling relationships between entities;
- an *integrated platform* called *HIGGINS* (Human Intelligence Games Engine), that instantiates KA games with interesting HITs by generating questions and multiple-choice answer candidates;
- an IE component that utilizes statistical LM's, combined with semantic resources for relational phrases;
- an HC component that employs ranking and diversification based on the IE-derived statistics;
- experiments that demonstrate the benefits of combining IE and HC.

II. HIGGINS System Architecture

Basic HITs. Our Higgins system provides the engine, data, and procedures for gathering relational knowledge through human-intelligence games (see Fig. 1). Specifically, it aims to collect, confirm, or assess instances of binary relations between named entities. In this paper, we focus on relations between people in the context of movies, books, biographies, or real-world events in the news. The *basic HIT unit* is to present the user with a pair of entities and a brief text about the context, and ask the user to provide the relation that holds between the two entities. This task has various ramifications:

1) the relation is unknown and we want a human to choose it from a set of suggestions;
2) the relation is unknown and we want a human to provide it by typing a short text phrase;
3) the relation is known and we want a human to confirm it;
4) we have a hypothesis about the relation and we want humans to assess its certainty.

In this paper, we mostly focus on cases 1 and 2, as they are the most demanding ones from a crowdsourcing perspective.

Advanced HITs. The above notion of HITs can be generalized by considering all three elements of a relational instance as either given by the system or to be filled in by the human. For example, we could provide the user with an entity and a relation and ask for the second entity. *"Michael Corleone" "order to kill" ?x* would be a concrete example where the question mark indicates the unknown slot. Even more generally, we could consider the context of such *knowledge triples* as a variable in the game as well. For example, when we are interested in relationships among the characters of a movie or in relationships among politicians in a news story, we could provide one or more relational instances and let the user guess the movie or event reported in the news article. To make this interesting, we could anonymize one or more of the entities. For example, we could show to the user: *"!Bambi" "order to kill" "!Dumbo"* and *"!Bambi" "brother of" "!Dumbo"* where the exclamation mark denotes an anonymized entity. Movie aficionados would guess that the context is the movie "Godfather Part 2".

Definition. A *knowledge triple* is a triple $\langle e1, r, e2\rangle$ consisting of entities $e1$ and $e2$ and a relation r, where each of the three slots may also be a variable (starting with a question mark and indicating a missing value). A *knowledge quad* is a quadruple $\langle c, e1, r, e2\rangle$ where c is a *context entity* and the other three components form a knowledge triple within the scope of c. Again, all slots may also be variables. In both triples and quads, we allow *semantic classes* in the place of the $e2$ entity. In this case, r is the *type*relation (an example being *"Michael Corleone" type MafiaGangster*).

Game Structure. A *knowledge acquisition game* generated by the Higgins system is a *sequence of interactions* between the system and one or more players, centered around the notion of knowledge triples or knowledge quads (if context is included). In each interaction, the system generates a knowledge triple or quad, presents it to the player(s), and asks the player(s) to complete the missing slot values (denoted as variables above).

An acquisition task transforms a partially bounded quad instance, generated by the IE engine, into a fully bound instance by completion of all bindings via inputs from the human players. The players choose from the provided list of candidates or type-in their response. The following is a (non-exhaustive) list of

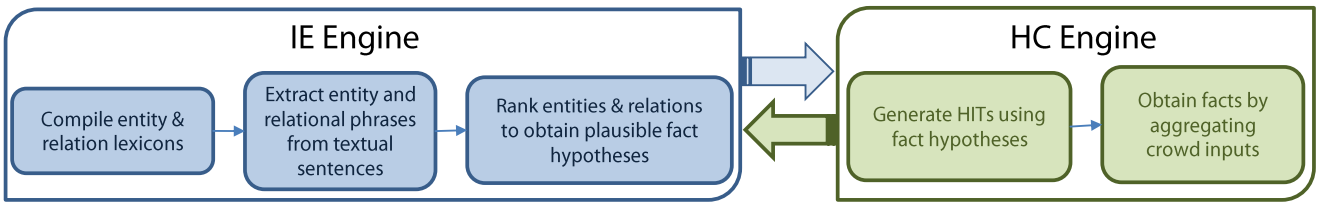

Fig. 1. System Architecture

different types of HITs depending on the bound (in boldface) and unbound arguments (in italics):

- $\langle \mathbf{c}, \mathbf{e1}, \mathbf{e2}, r \rangle$
  Example: In the context of the **movie Godfather**, how were **Michael Corleone** and **Sonny Corleone** *related*?
- $\langle c, \mathbf{e1}, \mathbf{e2}, \mathbf{r} \rangle$
  Example: In which *movie* did **Lord Aragorn marry Elf-Queen Arwen**?
- $\langle \mathbf{c}, \mathbf{e1}, e2, \mathbf{r} \rangle$ and $\langle \mathbf{c}, \mathbf{e2}, e1, \mathbf{r} \rangle$
  Example: In the **1992 presidential elections**, *whom* did **Bill Clinton defeat**?
- $\langle \mathbf{c}, e1, e2, \mathbf{r} \rangle$
  Example: In **JK Rowling's book Harry Potter and the Half-Blood Prince**, *who* **killed** *whom*?

This basic game structure can be extended in several ways:

- *Anonymized slots:* The $e1$ and $e2$ slots in a knowledge quad may be anonymized (denoted by an exclamation mark) and consistently used across multiple interaction steps of a game.
- *Multiple players:* Triples or quads can be shown to a single player or to multiple players simultaneously. In the latter case, we either aim to achieve agreement among players or we assign roles to players so that one player gives hints and the other player(s) attempts to guess the missing values.
- *Clues and callbacks:* Players may actively ask for clues; these would be provided by the system or another player. The clues are completely filled knowledge triples or quads: without variables, but possibly with anonymized slots.

System Perspective versus Player Perspective. Games are perceived from different perspectives by the system and by the players. The system understands the notion of knowledge triples and quads and how they can be built from an underlying database of entities, semantic classes, relations, and relation instances. Users, on the other hand, need to view these items in a simpler and intuitive form, either as natural language text with gaps to be filled or as form fields. Moreover, humans may refer to entities in shorthand form or by role names rather than using "official" entity names, e.g., using "Tom" or "the family lawyer" instead of "Tom Hagen", or using "Lady Di" or "the queen of hearts" instead of "Diana, Princess of Wales". For relations, this kind of vocabulary diversity may be even stronger, e.g., by players typing "gave the order to eliminate" instead of "ordered to kill" or simply "killed".

Dictionary. To connect the system knowledge with the user vocabulary, the Higgins system constructs and uses an extensive and extensible dictionary of *surface names* for entities and *paraphrases* for relations. These are collected from a variety of knowledge bases and semantic resources: i) entities and their surface names from `en.wikipedia.org`, `freebase.com`, and `yago-knowledge.org`, and ii) relational phrases from WordNet [21], ConceptNet [41], ReVerb [19], and PATTY [32]. In the second category, WordNet and ConceptNet are (mostly) hand-crafted resources with high-quality collections of verbal phrases, their synonymy, hypernymy/hyponymy, and more. ReVerb and PATTY, on the other hand, are automatically built by mining relational phrases from Web corpora like ClueWeb or the full text of Wikipedia. The dictionary contains names for millions of entities, and several hundred thousands of relational paraphrases. In our experiments, we will focus, though, on a much smaller subset relevant for storylines of movies and books.

Statistics. Higgins uses the dictionary for generating questions and candidate-answers for game instances, with user-suitable surface forms for entities and relations. This entails mappings between phrases and, most importantly, ranking and diversification of phrases to be shown to users. All this is based on quantitative *relatedness weights*, automatically derived from *co-occurrence statistics* in large corpora. The weights reflect the prominence of names (e.g., "Lady Di" is more often used than "Miss Spencer") and the frequency, precision, and conciseness of relational phrases (e.g., "ordered to kill" is more often used and more easily interpretable than "gave the hit-man a ring"). For diversification, it is often important to understand and quantify that some phrases are near-duplicates in terms of their semantics. For an interesting game, multiple-choice questions to the user should often list only one or only a subset of highly related phrases (e.g., only one of "assassinated", "murdered", "killed", "eliminated", etc.). Relatedness weights are fed into statistical language models that drive our system in selecting knowledge triples and generating interaction steps during a game (see Section IV). Details on the statistical models are in Section V.

III. IE Engine

State-of-the-art methods for information extraction (IE) use rules and patterns, linguistic analysis, statistical learning, consistency reasoning, or combinations of these elements. To achieve high recall, our IE approach follows the rationale of *Open IE* [4], [19], [46], [51]: we aggressively gather noun phrases for entities and verb-centric or role-centric phrases for relationships. Given a corpus, e.g., a set of movie narratives, we process all its documents in three phases:

1. identifying entity occurrences,

2. gathering relational phrases from sentences that contain two interesting entity names, using light-weight NLP techniques like part-of-speech tagging,
3. pruning the set of potential candidate relationships, by using more expensive NLP techniques like dependency parsing.

Entity occurrences. We consider all proper noun phrases (names, used without article) as entities. In addition, we consider a specific set of general noun phrases and occurrences of personal pronouns. If possible, we relate these phrases to canonical entities registered in a knowledge base like Freebase or Yago. The phrase-entity pairs form the entity part of a dictionary.

More specifically, for our use-case of relationships among movie or book characters, we have analyzed the cast sections of about 3,000 Wikipedia articles on movies and also extensive descriptions of about 500 books in the sparknotes.com portal. We identified, using regular expressions, names of movie and book characters and their name variations: first names, last names, nicknames, role names, etc. This way, we have compiled an extensive dictionary of 70292 entity-mentions for 46240 entity-name pairs for movie characters. Examples are the following knowledge triples (all in the context of the Godfather movies): ⟨ *Vito_Corleone knownAs "Don Corleone"* ⟩, ⟨ *Santino_Corleone knownAs "Sonny"* ⟩, ⟨ *Kay_Adams knownAs "Michael's girlfriend"* ⟩, ⟨ *Virgil_Sollozzo knownAs "the Turk"* ⟩, ⟨ *Tom_Hagen knownAs "the Consigliere"* ⟩.

In addition, we have identified general noun phrases near each character and mapped them to the type system of the Yago knowledge base, using the same heuristics that Yago uses for mapping Wikipedia category names onto WordNet synsets [44] and keeping only subtypes of the Yago class *person*. This way, we have compiled knowledge triples that provide us with fine-grained types of characters such as ⟨ *Virgil Solozzo, isa, drug dealer* ⟩. and ⟨ *Johnny Fontane, isa, singer* ⟩. This gives us a way of connecting role names occurring in narrative texts, e.g., "...the drug dealer", to the corresponding entities.

Finally, we have developed a heuristic method for resolving personal pronouns in such texts. We focus on pronouns in singular form, and we exploit gender information for characters. We map a male (female) pronoun to the closest preceding male (female) name occurring as grammatical subject. The gender properties for entities are derived from character descriptions using a classifier (over features such as frequencies of male vs. female pronouns, prior information on male vs. female first names, etc.). All these techniques serve to increase the number of sentences that we can tap into for extracting relationships. They produce errors at this IE stage, and the HC stage serves to capture these. Without these additional steps, the number of sentences that contain two explicitly and fully named entities and an interesting relation would often be disappointingly low. Movie plots in Wikipedia, for example, contain long and complex sentences with intensive use of pronouns and role names.

These techniques can be applied, with minor variations, to other Web sources such as imdb and other movie portals, and can be generalized to other kinds of entities. For example, for people's biographies as context entities, we can extract surface names of related entities from href anchor texts in Wikipedia and other sources.

Relational phrases. As surface expressions for binary relations, our system uses lexico-syntactic patterns like verbal phrases occurring in sentences that mention two entities. For mining these, we give ourselves a head start by using verbs from WordNet [21] and ConceptNet [41], and more general phrases from ReVerb [19] and Patty [32].

These semantic resources already provide us with a fairly big set of phrases. However, when considering human relationships in movies or books, one sometimes comes upon very special phrases. Therefore, we have additionally mined paraphrases for a manually compiled set of 500 relations, by mining patterns and their co-occurring entity pairs in the ClueWeb'09 corpus. The methodology closely follows that used for constructing Patty [32], but it is applied here to a specifically chosen set of relations relevant for storylines of movies and books. We limit phrases to one of the following two cases: 1) verb constructs or verbal phrases ending with a preposition (e.g. "falls in love with"), or 2) noun phrases ending with a preposition (e.g., "wise counselor of"). Examples of knowledge triples with phrase-relation pairs obtained in this way are: ⟨ *"had an affair with" means romanced_with* ⟩, ⟨ *"stabbed from behind" means attacked* ⟩, ⟨ *"cut his head off" means murdered* ⟩.

We have a total of 116000 relational phrases compiled in this way. We also use the co-occurrence frequencies of phrases with entity pairs in the ClueWeb'09 corpus to estimate the *semantic relatedness* between phrases, which is later used by the HC engine for ranking and diversification in questions and candidate answers. This statistical estimation is discussed in detail in Section V.

Pruning of candidates. By detecting entity mentions in sentences and matching phrases from the dictionary in the same sentences, we can compile a huge set of candidates for relationships between entities. This gives us a very rich pool for generating HITs at the HC stage. However, the ratio of spurious relation instances is very high. This is due to the complexity of the input sentences and the aggressiveness of our IE methodology. We experimented with more conservative approaches as well, but they tend to miss out many interesting candidates and lose big in recall.

To reduce the number of false positives, we employ a more expensive post-processing phase on the collected relationship candidates by running a dependency parser on all relevant sentences. Specifically, we use the Stanford Parser [14] and identify all dependency paths in a sentence connecting a pair of entities that occurs in a candidate from the previous phases. If no such path exists, the candidate is discarded. Otherwise, we retain all candidates whose relational phrases overlap with words on at least one of the dependency paths. For example, in the sentence "Le Chiffre abducts Vesper Lynd and uses her as bait to capture James Bond.", to detect the relation between Le Chiffre and James Bond, we keep the top-k relations that match the dependency path `[abducts uses as bait to capture]`.

IV. HC Engine

The IE engine generates *hypotheses* $h = \langle c, e1, e2, r \rangle$. Hypotheses are fed into HITs generated by the HC engine. When the players fill in values for the unbound arguments, the

hypotheses turn into fully bound knowledge quads. Thus, a set of HITs H is eventually turned into a set of knowledge quads K. HITs come in two flavors. If instance $h \in H$ contains unbound arguments, the HIT is termed as an *acquisition task*. The player's objective is to come up with the correct binding(s). The IE engine may additionally provide candidate values for unbound arguments, in which case, the player sees a *multiple-choice question* for the correct binding. If the instance $h \in H$ contains only bound arguments, the HIT is called a *validation task*. Here the player's objective is to confirm or refute the hypothesis.

Question Generation. The key issue for the HC engine is to generate the questions in a HIT and, if it is a multiple-choice question, a set of candidate answers. For simpler explanation, we focus on the case of hypotheses where only the relation argument is unbound.

In principle, we could randomly pick instances from the pool H of hypotheses generated by the IE engine. However, this would inevitably yield poor results, because many instances would not be suited for a specific user. Examples are entities that are unfamiliar to the user and/or obscure entities involving peripheral characters in a movie or book.

To avoid this pitfall, we tailor the questions to i) relate to context entities that match the user's interests (e.g., movies she knows well), and ii) refer to relations between salient entities (e.g., main characters) of popular movies or books. For the latter, we exploit knowledge sources like Wikipedia, imdb.com, and sparknotes.com.

Once a hypothesis $h \in H$ is chosen as a HIT for a particular user, it merely needs to be cast into the surface form that the game uses, e.g., by adding cues from text about the context entity and the sentence that led to h and its surrounding paragraph.

Candidate Answers Generation. In a game, our system could simply ask the user to fill in the missing value for a relationship, by providing a free-text form field. However, experience shows that it is easier to engage a broad set of users with multiple-choice questions. Therefore, we generate a small number of candidate answers (usually 5) and additionally offer a free-text field for entering other relationships. For the game effectiveness, it is important that the offered candidate answers i) are reasonable, that is, exclude nonsense relations for the given context entity (e.g., exclude "is the grandmother of" for a question about James Bond and Le Chiffre), ii) include a good answer, if known from the original sentence that generated the HIT, and iii) are sufficiently diverse so that users see actual choices, as opposed to proposing only candidates that are so close that only a very sophisticated user could distinguish them (e.g., "orders to kill" vs. "hires someone to assassinate" - which would be considered the same by most players).

To satisfy these desiderata, the possible answers for a HIT are first ranked, based on the statistical language model presented in Section V, and then diversified:

- **Ranking:** For this step, we start with the relational phrase from the original sentence that generated the HIT question. We use the statistical language model for phrase relatedness (see Section V to enumerate semantically similar phrases in descending order of relatedness scores. For example, when starting with "falls in love with", highly related phrases would be "loves", "love affair with", "romantic passion for", "engages in romance with", etc.
- **Diversification:** For diversification, we aim to remove near-duplicates from the top-k candidates according to the relatedness ranking. To this end, we remove all phrases that have significant overlap with a phrase that is ranked higher. Overlap in words is considered significant if there is a noun or verb contained in both phrases. In the above example, the phrases "loves" and "love affair with" are viewed as near-duplicates. Ditto for 'romantic passion for", and "engages in romance with". In addition to this syntactic form of diversification, we use randomization by picking the final phrases for the multiple-choice question from a larger pool of top-ranked and de-duplicated candidates (typically picking 5 out of a pool of 20).

Game Playing and Player Rewards. Our system uses predefined question templates and plugs in the context, the entities, and the relation(s), to present a HIT to a player. The player can choose to either respond or skip a question that is presented. Upon providing a response, game points are awarded to the player. Notably, the game can be configured to give additional cues (from the context) upon request by the player. This is strategic: when the presented choices do not satisfy the player, the player can see the sentence for which the entity-relation fact occurs. In this way, noisy relations (that may occur for instance due to errors in pronoun resolution) can be corrected by players. To do this, players simply engage human intelligence using which correct resolution is much easier. And, we enhance our KB quality using the intelligence of humans who may not be experts in specific movies or books.

The HC engine has no gold standard solutions to directly evaluate the correctness of the players' responses. In a single-player environment where players play the game independently, we deal with this issue by awarding *future agreement bonuses* based on the fraction of the players that agreed with the player's responses. The objective of each player thus is to maximize her long-term overall score and obtain a top rank among all players. And the objective from the system's point of view is to place more faith in the highly-agreed responses, as we expect that these will yield a better KB.

V. Phrase Dictionary and Statistical Language Models

The generation of candidate answers for multiple-choice questions is based on phrases from our dictionary, and ranking and diversifying them given a specific question (HIT). Recall that the dictionary is built from a variety of sources, including both semantic resources like WordNet or ConceptNet and statistics collected by Web mining. For each pair of relational phrases, we compute a *relatedness score* using a mixture model that combines multiple kinds of statistical language models (LM's).

Language Models. LM's are a principled model in information retrieval (IR) for ranking in different contexts [53]: ranking documents for keyword queries, ranking passages as a first stage in question answering, ranking entities for semantic search, ranking sentences for summarization, etc. Such advanced IR models are not widely known outside the IR community and

rarely used for problems other than IR tasks. Our approach here is a novel way of harnessing these IR models for knowledge acquisition.

An LM is a probability distribution over words or bigrams or variable-length sequences, often using a multinomial model. Each document, each query, each passage, etc. is associated with an LM, where the parameters are estimated from the document, query, or passage, respectively, and are additionally smoothed by statistics over an entire corpus or query log or other background information. The principle for ranking is that the best document for a given query is the one whose LM has the highest likelihood of generating the query (and analogously for other IR tasks). This is typically cast into computing conditional probabilities $P[q|d]$ or information-theoretic scores like the Kullback-Leibler divergence $KL(q|d)$ (aka. relative entropy).

LM for Phrase Relatedness. For our setting of phrase relatedness, a specifically relevant form of LM is *translation models* [5], used in IR for cross-lingual retrieval and tasks where one vocabulary must be translated into a different one. For example, a French query leads to ranking English documents, or a medical layman's query needs to be evaluated against a corpus of health-related documents with a different terminology. Such settings typically consider probabilities of the form

$$P[q|d] = \sum_{w \in q} \left(\sum_{v \in d} P[w|v] P[v|d] \right).$$

The $P[w|v]$ terms constitute the translation model.

In our setting, we use this principle in a generalized way for "translating" one relational phrase into a different paraphrase. For example, we are interested in $P[\text{“}romance\ with''|\text{“}loves'']$. This is the basis for the phrase relatedness scores. The details of this high-level idea lead to a novel form of LM for knowledge acquisition, as discussed in the following.

Parameter Estimation. Our approach for estimating the probabilities $P[w|v]$ for phrases w and v uses co-occurrence statistics that related phrases with entity pairs. For each phrase p, we gather from the ClueWeb'09 corpus pairs of noun phrases that p occurs with. This yields a *support bag* $Sbag(p) = (n1, n2)|\langle n1, p, n2\rangle \in corpus$ for p. We liberally use noun phrases here, rather than restricting ourselves to individual entities, to capture also widely used expressions such as *"Vesper ... romance with ... her lover"*. We use lemmatization tools to normalize noun phrases, e.g., collapsing singular and plural forms.

We could now use mutual information or similar measures to model the relatedness between phrases. However, given the sparseness and noise in the underlying Web corpus, some initial experimentation showed that simpler techniques actually gave more robust results. So we actually adopted the Jaccard coefficient as a basis for estimating relatedness, and set $P[p|r]$ to $\frac{|Sbag(p) \cap Sbag(r)|}{|Sbag(p) \cup Sbag(r)|}$. Since we use bags (rather than sets), where occurrence frequencies of the same noun-phrase pairs matter, this is equivalent to a weighted variant of the Jaccard coefficient,

$$J(r,p) = \frac{\sum_x (min(freq(x \in Sbag(p)), freq(x \in Sbag(r))))}{\sum_x (max(freq(x \in Sbag(p)), freq(x \in Sbag(r))))}.$$

Mixture Model. Since we combine these statistics with semantic resources for phrases, we need to combine different kinds of relatedness scores as well. For resources like WordNet or ConceptNet, there are explicit, manually crafted relations that connect phrases. In WordNet, two verbs or phrases can be *synonymous*, or one could be a *hypernym* (generalization) or *hyponym* (specialization) of the other, or they could be connected by a *derivational* relation (e.g., the verb "assassinate" is related to the noun "assassination", which in turn has hypernyms etc.). In ConceptNet, there are explicit relations *isA*, *SimilarTo*, *DerivedFrom*, RelatedTo, and *ConceptuallyRelatedTo*. We associate with each type of relation between phrases a coefficient set to a fixed value between 0 and 1. For example, when two phrases are in the *DerivedFrom* relation in ConceptNet, we assign a fixed relatedness score of $\gamma_{DerivedFrom}$. All these scores are combined into a mixture-model score, first for WordNet and ConceptNet separately, and finally combined with the LM-based statistical scores from above. So the final relatedness score between phrases p and r becomes:

$$\begin{aligned} relscore(p,r) = {} & \alpha_{stats} \times score_{stats}(p,r) \\ & + \alpha_{WordNet} \times score_{WordNet}(p,r) \\ & + \alpha_{ConceptNet} \times score_{ConceptNet}(p,r). \end{aligned}$$

VI. Experiments

We study the *quality of results* obtained from crowds and the *costs* incurred. We compare Higgins against baseline methods that rely solely on IE only or HC only. We also compare the effect of statistical and semantic components as components of our Higgins engine, and we look at the robustness of our methodology by studying different kinds of input data.

A. Setup

We conducted experiments using plots and character descriptions of books stories from SparkNotes and movies articles from Wikipedia. We extracted the character roles from the description of the movie/book articles and resolved their occurrences in the plot sentences using the techniques described in Section III. We compiled data for about 3,374 movies and 352 books, with a total of 52,700 characters. We generated about 27980 questions for the game.

To deal with the complexity of long sentences in the narrative texts, we used dependency parsing and extracted the shortest paths between occurrences of the movie/book characters. All words on the these paths were used to fetch possible relational phrases from the dictionary of relations described in Section III. For question generation we considered all sentences that mention the pair of character roles and used the statistical language model to rank and provide a short list of top-5 diversified candidates.

Obtaining aggregate candidates. If $S = \{s_1, s_2, ..s_n\}$ are the sentences that contain mentions of the character pair, we rank the relational phrases from the dictionary D using the mixture model (described in Sec. V) for each sentence s_i and obtain the ranked lists $R = \{L_1, L_2, .., L_n\}$ respectively. The aggregate score for each relation $r \in D$ is obtained as

EXAMPLE 1

Book: The Kite Runner
Character One: Assef
Character Two: Sohrab
Sentences: *"Sohrab threatens Assef with his slingshot and when Assef lunges at him, Sohrab shoots him in the eye, allowing Amir and Sohrab to escape."*
o Sohrab *threatens* Assef
o Sohrab *shoots* Assef
o Sohrab *lunges at* Assef
o Sohrab *kills* Assef
o Sohrab *finds* Assef

EXAMPLE 2

Movie: Lord of the Rings: Return of the King
Character One: Frodo Baggins
Character Two: Gollum
Sentences: *"Gollum betrays Frodo, leaving him in the lair of giant spider Shelob."*
"When Sam and Frodo are captured by the Rangers of Ithilien, Frodo reveals Gollum's presence to spare his life; Gollum nevertheless feels betrayed and begins plotting against the new master."
"Frodo and Sam take pity on him, understanding the burden of the ring."
o Frodo Baggins *was captured by* Gollum
o Frodo Baggins *was forced to endure* Gollum
o Frodo Baggins *made life miserable for* Gollum
o Frodo Baggins *left* Gollum
o Frodo Baggins *betrays* Gollum

Fig. 2. Sample HITs

$\sum_{i=1}^{n}(1/pos_i(r))$, where $pos_i(r)$ is the position of r in the ranked list L_i. The relations are ranked by the aggregate score and the top-5 are presented as choices to the player.

These top-5 options always include the most likely candidate phrases. However, because of errors in dependency parsing, pronoun resolution, and other steps, even the most likely phrase could be incorrect. For some sentences and choice of entity pairs, actually no relationship would hold. Therefore, the interface for HC contributors always included the options NONE and OTHER. In the latter case, the users could enter a relationship as free text. Furthermore, for some entity pairs, multiple relationships could hold and the top-5 candidates could include more than one of these. Fig. 2 shows two example questions generated by the system and the aggregated answer candidates.

For evaluation, we consider two samples drawn from the pool of all questions: a "prominent" set of questions drawn from popular movies & books, and a "random" set of questions drawn randomly from the entire pool (excluding the questions from the prominent sample). We did this for two different choices of input texts: i) plots and story summaries of movies and books (shown as MoviePlots & BookPlots respectively), and ii) descriptions of main characters in the movies or books (shown as MovieCast & BookCast respectively). Table I summarizes the test samples.

Crowdsourcing Setup. When deploying the game on CrowdFlower, as there is no provision to swap the character roles to allow we generated twice the number of options in a question by considering both orders of the character roles. Each HIT also shows the sentences from which the options were generated to the human worker. Each question is evaluated by 5 different workers on CrowdFlower.

B. Quality and Costs of HIGGINS Results

This section presents results on the precision and recall of acquiring relationships with Higgins, as well as insights on inter-judge agreement and the overall cost. All results here are from the full configuration that integrates all assets described in the paper. The mixture-model parameters (see Section V) were set to $\alpha_{stats} = 0.991$, $\alpha_{WordNet} = 0.0049$, $\alpha_{ConceptNet} = 0.0039$.

We manually constructed ground-truth answers for the two datasets. Since some sentences produce multiple relationships, we consider all correct answers for the ground-truth. For character role pairs that do not describe any relation in the sentences, we consider NONE as the ground-truth. In cases where a relation exists in the sentence but is not captured by any of the generated answers, OTHER is considered as ground-truth. This naturally leads us to two different settings for evaluating the HC results. In the **conservative** evaluation, we only consider questions from the ground-truth that do not have NONE or OTHER as answers, thereby evaluating the goodness of the options generated by our system. In the **liberal** evaluation, we calculate precision and recall over all the answers including NONE and OTHER.

Precision. We define the notion of precision to account for multiple possible answers for a question. Let $Q = \{q_1, q_2, ..q_i, ..q_n\}$ be the set of generated questions. For each q_i we have $\{f_1, f_2, ...f_j, ..f_{12}\}$ options provided to the players, where $\{f_1, .., f_5\}$ are facts obtained top-5 candidate answers and $\{f_6, .., f_{10}\}$ are obtained by swapping the entity roles. f_{11} and f_{12} are the NONE and OTHER options, respectively. For each $f_j \in q_i$ we have,

$$f_j = \begin{cases} k & \text{count of workers who chose } f_i \\ 0 & \text{otherwise} \end{cases}$$

Let t_j be a possible answer to question q_i. We have $t_j \in \{0, 1\}$ where $t_j = 1$ if the answer is in the ground-truth of correct answers, and 0 otherwise. The precision P is calculated as $P = \sum_j (f_j * t_j) / \sum_j f_j$. The overall precision we report is the average of P over all questions in the set Q.

Recall. Recall measures the fraction of correct answers in the ground-truth that the HC game obtained. For question q_i and answer t_j we set $g_j = 1$ if either i) at least one player chose t_j or ii) the majority of workers who answered q_i chose a correct t_j. We refer to the first option as *Recall-any* and to the second as *Recall-majority*. The recall per question q_i then is $R = \sum_j (t_j * g_j) / \sum_j t_j$. The overall recall is averaged over all questions q_i in Q. Note that our setting allows multiple correct answers per question. Therefore, our notion of majority allows that different users choose different answers as long as all of them are correct.

The results are shown in tables II, III and IV. We see that the system generally achieved high precision and, even in the majority mode, decent recall.

Regarding the cost of crowdsourcing, we looked at the inter-judge agreement as a function of the number of contributors for a question. We broke this down by the number of workers. The results reflected by the Fleiss Kappa measure and the average

TABLE I
DATASETS FOR EVALUATION

	Prominent Sample		Random Sample	
	No. of movies/books	No. of questions	No. of movies/books	No. of questions
MovieCast	20	109	59	75
MoviePlots	20	130	58	75
BookCast	20	135	49	75
BookPlots	20	167	61	75

TABLE II
PRECISION MEASUREMENTS FOR HIGGINS

	Prominent Sample		Random Sample	
	Conservative	Liberal	Conservative	Liberal
MovieCast	0.661	0.795	0.732	0.848
MoviePlots	0.558	0.678	0.633	0.715
BookCast	0.600	0.754	0.529	0.699
BookPlots	0.695	0.781	0.601	0.723

TABLE III
RECALL MEASUREMENTS FOR HIGGINS WITH PROMINENT SAMPLE

	Prominent Sample			
	Conservative		Liberal	
	Recall-Any	Recall-majority	Recall-Any	Recall-majority
MovieCast	0.673	0.578	0.963	0.862
MoviePlots	0.625	0.616	0.927	0.880
BookCast	0.633	0.578	0.956	0.867
BookPlots	0.743	0.663	0.948	0.855

TABLE IV
RECALL MEASUREMENTS FOR HIGGINS WITH RANDOM SAMPLE

	Random Sample			
	Conservative		Liberal	
	Recall-Any	Recall-majority	Recall-Any	Recall-majority
MovieCast	0.727	0.720	0.961	0.950
MoviePlots	0.711	0.547	0.978	0.840
BookCast	0.591	0.520	0.921	0.853
BookPlots	0.689	0.603	0.971	0.850

precision are shown in Table V. We see that the agreement increases with more users per questions, but quickly saturates. So for high-quality HC contributions, we do not need a large number of users per question. Of course, this holds when the questions and candidate answers are generated in a meaningful way. This is exactly the contribution of our IE engine for a well-prepared and cost-effective HC phase.

TABLE V
INTER-JUDGE AGREEMENT

	Prominent			Random		
No. of workers	**3**	**4**	**5**	**3**	**4**	**5**
Fleiss Kappa	0.470	0.470	0.460	0.510	0.500	0.510
Avg. Precision	0.722	0.748	0.742	0.729	0.739	0.739
Avg. Recall-Any	0.843	0.907	0.914	0.863	0.910	0.950

C. Comparison of HIGGINS against IE-only and HC-only Baselines

Our main hypothesis in this paper is that the combination of IE and HC yields synergies and thus benefits that neither an IE-only nor an HC-only method can achieve. For experimentally insight in this hypothesis, we compared Higgins against two baselines that use solely IE or solely HC.

1) IE-only Baseline: OLLIE [31] is a state-of-the-art Open IE system that performs bootstrap learning of patterns on dependency parse trees to detect factual triples. The OLLIE extractor further analyzes the structure of the dependency trees enabling it to detect non-factual assertions such as beliefs, hypothetical or conditional sentence constructs. We ran OLLIE on our datasets and manually identified all correct extractions computed by this IE-only method. Since OLLIE produces relations directly from the textual input while Higgins presents the relations from its dictionary of relations, we cannot easily define a notion of recall to compare the two systems. Even the comparison based on the standard notion of precision would be debatable or unfair in this setting, as each system may capture relations in different representations. Rather than using heuristics for thresholding on extraction confidence measures, we compared the set of OLLIE extractions directly against the extractions by Higgins and vice versa. This is done as follows.

Let $H \subseteq Q$ be the set of questions for which Higgins provided at least one truly correct answer. Let $I \subseteq Q$ be the set of questions for which OLLIE generated a correct fact. Now we can compare the performance of OLLIE relative to what Higgins returned, and vice versa. Table VI shows values for $|H|$, $|I|$, $|H \setminus I|$, $|I \setminus H|$, and $H \cap I$ respectively.

The numbers in Table VI clearly show that Higgins provides many more correct facts, compared to OLLIE (shown by $|H \setminus I|$). Conversely, OLLIE provided only very few correct facts that were not acquired by Higgins (shown by $|I \setminus H|$). This shows that the joint IE-and-HC method of Higgins outperforms state-of-the-art IE-only methods.

2) HC-only Baseline: As a baseline for an HC-only method, we generated candidate answers for questions in an uninformed random manner but using heuristics to avoid meaningless candidates. For each pair of character roles in a question, we picked the pool of candidates from the dictionary of relations that have a non-empty word-level Jaccard overlap with the words on the dependency path between the characters. For example, A sample of five relations chosen randomly from this pool were presented to the worker. The precision and recall measurements are summarized in tables VII and VIII.

TABLE VI
COMPARISON OF HIGGINS (H) AND OLLIE (I)

	Prominent					Random				
DataSet	$\|H\|$	$\|I\|$	$\|H \setminus I\|$	$\|I \setminus H\|$	$\|H \cap I\|$	$\|H$	$\|I\|$	$\|H \setminus I\|$	$\|I \setminus H\|$	$\|H \cap I\|$
MovieCast	76	18	61	3	15	58	27	33	2	25
MoviePlots	81	30	57	6	24	54	16	40	2	14
BookCast	100	45	61	6	39	47	10	38	1	9
BookPlots	128	48	84	4	44	51	19	34	2	17

TABLE VIII
RECALL MEASUREMENTS FOR HC-ONLY METHOD

		Prominent Sample		Random Sample	
Dataset		Conservative	Liberal	Conservative	Liberal
MovieCast	Recall-Any	0.822	0.953	0.78	0.960
	Recall-Majority	0.355	0.383	0.236	0.272
MoviePlots	Recall-Any	0.674	0.952	0.635	0.898
	Recall-Majority	0.116	0.193	0.148	0.202
BookCast	Recall-Any	0.895	0.926	0.706	0.800
	Recall-Majority	0.118	0.119	0.400	0.480
BookPlots	Recall-Any	0.709	0.962	0.640	0.902
	Recall-Majority	0.206	0.255	0.173	0.200

TABLE VII
PRECISION MEASUREMENTS FOR HC-ONLY METHOD

	Prominent Sample		Random Sample	
	Conservative	Liberal	Conservative	Liberal
MovieCast	0.316	0.601	0.196	0.594
MoviePlots	0.155	0.508	0.118	0.459
BookCast	0.111	0.700	0.430	0.708
BookPlots	0.316	0.583	0.149	0.490

D. Impact of HIGGINS Components

Higgins uses a variety of components, and we were interested in identifying which components are essential for high-quality output and which ones merely contribute marginal benefits. We studied different configurations of Higgins, and how this influenced the generated candidate answers presented to the user and how this in turn influenced the overall quality of the HC stage. We compared the *full* system against two variants that used i) only the semantic components (WordNet and ConceptNet) for the statistical model, or ii) only the statistical assets (excluding WordNet and ConceptNet). These two variants are referred to as *sem-only* Higgins and *stat-only* Higgins, respectively.

The results over the random sample are shown in Tables IX and X for the precision and recall, respectively. In all cases, we see that the *full* combination outperforms the other configurations by a large margin. This demonstrates that combining semantic resources and statistics is vital for generating meaningful answer candidates and obtaining high-quality results.

E. Discussion: Strengths, Limitations, Lessons Learned, and Outlook

As the above results substantiate, Higgins delivers high performance. It consistently and significantly outperforms both the IE-only and HC-only approaches in terms of recall and/or precision. Perhaps even more encouraging is the result that high inter-annotator agreement per question can be reached, even with only a small number of participants. This is crucial, as it implies that the high-quality output of Higgins can be achieved at low capital costs (which are a typical concern whenever crowdsourcing is involved).

Limitations: One would expect that HC-only methods can achieve near-perfect results as the number of workers per question is increased, thus aggregating the "wisdom of crowds" over more independent people. Apart from the fact that this may quickly become prohibitively expensive, our experience with a small subset of questions even makes us skeptical whether the user inputs actually converge to an agreed-upon fact in (nearly) all cases. Some relationships between characters in movies or books are very subtle, and the sentences that express them in narrative descriptions are sometimes quite sophisticated. An example is: "*With a single gunshot, Blondie severs the rope, dropping Tuco face-first onto his share of the gold. Blondie smiles and rides off as Tuco curses him in rage, shouting, 'Hey Blonde! You know what you are? Just a dirty son of a bitch!'*", from the movie "Good, the Bad, and the Ugly". Should this lead to a fact ⟨*Blondie helped Tuco*⟩ or ⟨*Blondie saved Tuco*⟩ or ⟨*Blondie likes Tuco*⟩ or ⟨*Tuco likes Blondie*⟩? Such situations pose a daunting challenge to both humans and automated IE. It is widely open whether a pure HC method or a combined IE & HC approach are suitable for such difficult inputs.

IE for HC: All our experiments confirmed that, in most cases, having a smart IE phase before embarking on HC is very beneficial. The quality of the candidate answers generated for multiple-choice questions is substantially improved by our

TABLE IX
IMPACT OF HIGGINS VARIANTS: PRECISION MEASUREMENTS

	Random Sample					
	stats-only		sem-only		full	
Dataset	Conservative	Liberal	Conservative	Liberal	Conservative	Liberal
MovieCast	0.605	0.754	0.713	0.797	0.732	0.848
MoviePlots	**0.252**	0.536	**0.299**	0.516	**0.633**	0.714
BookCast	0.544	0.630	0.477	0.649	0.529	0.699
BookPlots	**0.254**	0.611	**0.318**	0.456	**0.601**	0.723

TABLE X
IMPACT OF HIGGINS VARIANTS: RECALL MEASUREMENTS

		Random Sample					
		stats-only		sem-only		full	
Dataset		Conservative	Liberal	Conservative	Liberal	Conservative	Liberal
MovieCast	Recall-Any	0.623	0.888	0.663	0.878	0.727	0.951
	Recall-Majority	0.623	0.836	0.720	0.875	0.729	0.960
MoviePlots	Recall-Any	0.339	0.868	0.401	0.795	0.711	0.978
	Recall-Majority	**0.240**	0.613	**0.284**	0.595	**0.547**	0.840
BookCast	Recall-Any	0.528	0.821	0.514	0.918	0.591	0.920
	Recall-Majority	0.513	0.716	0.534	0.794	0.520	0.853
BookPlots	Recall-Any	0.260	0.948	0.399	0.872	0.689	0.971
	Recall-Majority	**0.173**	0.640	**0.253**	0.547	**0.603**	0.850

combination of statistical (language-model-based) and semantic (ontology/thesaurus-based) assets. Although the high-level idea itself may be relatively obvious and some of the underlying blocks may be easy choices, we believe that our judicious combination of the right assets and controlling their interplay is one of the contributions of the Higgins project.

HC for IE: In addition to IE boosting the performance of HC, there is also a virtuous feedback from HC to IE. As we acquired more and more correct relationships from the HC stage of Higgins, we can use these to improve the semantic dictionary and the language-model statistics used in IE to drive the generation of candidate answers. Most notably, the translation LM that estimates the relatedness of different phrases is crucial. Texts sometimes use subtle or rare formulations (sometimes even with irony) to denote a relationship. For example, phrases like "develops feelings for" or "is drawn closer to" are rare, so their relatedness with "falls in love with" cannot be estimated that well by IE techniques alone. Once we have HC results for such cases, we can feed them back into the IE statistics and improve the estimates of the translation LM's.

Cost of Crowd-sourced Experimental Research: In our experimental studies, we tried to scale up as much as possible. We realize, though, that we did not reach out to the level of large game communities or big online groups. We believe that our approach has the potential for further scaling, but at this point, the total cost of experiments has been a limiting factor. In total, our experiments involved more than 12800 HITs on CrowdFlower. Although each HIT costs only 5 cents, the total cost is a concern. In addition and even more troublesome, we had to create ground-truth output for all instances that were evaluated. In total, these were more than 2,000$*$10 question-answer pairs. We are not aware of similarly intensive studies on crowdsourcing for knowledge acquisition. All our experimental data is made available for further research in the community at http://www.mpi-inf.mpg.de/yago-naga/higgins/.

Outlook: Higgins manages to produce high quality knowledge at low crowd costs. This begs the interesting question, at least from a theoretical point of view: If money were not an issue, how much can we improve our KA engines, say by engaging a much greater number of users per question? Put differently: is gathering the "wisdom of the crowds" by engaging large numbers of humans, capable of improving substantially the quality of KA? Perhaps surprisingly at first sight, our preliminary evidence is not supportive of this. The complex relationships we are seeking to discover are by their very nature very subtle and they typically appear in narratives in various sophisticated ways. As such, they typically escape the attention of many. Intrigued by this observation, our future plans include the study of robust KA engines (for our problem domain space) which will be able to predict, subject to cost constraints, how extensive HC involvement is needed and what will be its expected quality improvement/impact.

VII. RELATED WORK

Related research spans efforts in information extraction (IE) for knowledge acquisition (KA), crowdsourcing or "human computing" (HC), and combinations of them.

IE for KA. There is a wide variety of IE methods for knowledge discovery. These are based on rules and patterns (e.g., [6], [12], [20], [23] linguistic analysis (e.g., [19], [27], [33], [35]), statistical learning (e.g., [8], [10], [24], [38], [51], [54]), consistency reasoning (e.g., [34], [45]) and often combinations of all these elements. Methods and tools are typically customized to the characteristics of the input sources

for the IE process, ranging over a wide spectrum: infoboxes, categories and lists in Wikipedia; product-centric Web pages generated from Deep-Web databases; HTML tables from the Web; e-mails, news, and other texts with rich metadata; all the way to arbitrary natural-language documents from the Web or other text corpora. Large knowledge bases built this way contain millions of entities, organized into many thousands of semantic classes, and interconnected via hundreds of relationship types.

Despite these great advances, there is still a major problem regarding coverage, the underlying fundamental issue being the precision-recall trade-off. Conservative IE methods miss out on many interesting relationships, whereas aggressive methods exhibit a substantial drop in output quality and produce more noisy and incorrect results. To overcome the limited coverage of most IE approaches, the paradigm of *Open IE* has been proposed in [4] and further advanced in, e.g., [19], [38], [46], [51]. This is an aggressive approach which starts with the hypothesis that all noun phrases are entities and all verb-centric phrases are relationships. The hypothesis is further refined by patterns over part-of-speech tags, lexical information, and statistical arguments. Still, these methods produce a substantial amount of noise. Even variants that use dependency parsing on input sentences (e.g., [9], [31], [43]) cannot cope with complex sentences. Likewise, anaphora resolution (for pronouns etc.) often presents major obstacles.

HC for Various Tasks. With the advent of popular crowdsourcing platforms, like Amazon MTurk [2] and CrowdFlower [13], and the ensuing popularization of crowdsourcing techniques for a variety of applications, HC has gained massive attention. Academics in the areas of data and knowledge management have started principled research on how to systematically exploit collective human intelligence [17], [28]. The wide variety of HC applications includes quality assessment of rankings, summaries, and IE results [1], [17], [25], query answering [22], [36], [37], entity matching [16], [50], ontology alignment [39], image search and annotation [15], [47], [52], and acquiring commonsense properties [42], [49] and taxonomies [26].

Within data management, CrowdDB [22], Qurk [29], [30], and [36], [37] attempt to extend database systems by incorporating crowd functionality. CrowdDB augments the DBS with new data (that humans can easily find, say utilizing search engines) and aims to answer fuzzy-comparison queries, such as different text sentences or photos referring to the same entity. It models human participation in the task of query answering, extending SQL with operators reflecting these data-adding and comparison needs. Qurk [30] aims at managing the workflow and the returned answers by humans to HITs. It extends SQL and represents human answers as multivalued attributes and aggregation functions are defined (such as majority voting) to declare the eventual single attribute value. It also allows the addition of new operators, such as filters, sorts, and joins. For instance, join operators are employed whereby humans are called to find the same entities in different "tables" (e.g. identify the same persons in two different sets of photos). In [29] authors show how the human-based sort and join operators are implemented and optimizations are presented regarding the generation of HITs and their costs. Similarly, [37] attempts to involve humans in data management tasks, presenting a new query model which allows for predicates to be evaluated by humans, the DB, and/or external algorithms.

Crowdsourcing is increasingly being employed also for Information Retrieval tasks [1], [25]. CrowdSearch, [52], for example, tackles the problems associated with the low-quality results achieved when searching for images, especially when using mobile phones where image quality is typically poor. It adds a human-validation phase, whereby the AMT infrastructure is employed and human intelligence improves search result quality. Crowdsearch addresses the issues of selecting the results for which validation is sought and how to validate them, using models for the delay-accuracy-cost trade-offs involved in the crowdsourced tasks. Crowds have also been successfully employed for relevance assessments: [1] shows that Turkers can perform relevance assessment as well as TREC experts.

HC for KA. Finally, with respect to knowledge acquisition, crowdsourcing techniques also have enjoyed successes. An early representative example are common sense knowledge bases, with the Open Mind Common Sense (OMCS) project [40], being a prototypical example. OMCS relies on volunteers to provide statements of common sense regarding real-world objects, people, and events. In parallel to such 'brute-force' crowdsourcing efforts, games with a purpose, have shown that knowledge can be acquired, as a 'side-affect', when humans play carefully designed games. Verbosity [49] is an interesting example game, which was successful in terms of both the number of people players and the knowledge acquired. Interestingly, OMCS and Verbosity have been combined [42] in order to increase OMCS's 'entertainment value' and thus its human involvement. This is facilitated by the fact that the statements acquired by Verbosity resemble greatly the facts in ConceptNet [41] (OMCS's semantic network representation).

Very recently, crowdsourcing has also been employed for the creation of taxonomies [26], collecting seed facts as labeled training data [11], entity resolution [50], and ontology alignment [39]. However, all above efforts rely solely on human input for knowledge acquisition. Our architectural approach of integrating IE and HC has been sketched in a 2-page poster [55]; the current paper gives the full story on the HIGGINS architecture, its statistical and semantic assets, and our experiments with a real crowdsourcing platform.

VIII. Conclusion

This paper reported on the design, methods, and field testing of a novel system architecture that combines information extraction (IE) with human computing (HC) for advanced forms of knowledge acquisition. We successfully applied this combination to the difficult task of compiling relationships between characters of movies or books. By harnessing semantic and statistical resources, the IE engine is able to create meaningful questions and answer candidates, fed into the HC engine for crowdsourcing. This way, the cost of the HC phase can be controlled and is substantially reduced compared to an HC-only approach. Our experiments demonstrate the benefits of our system in terms of both output quality and HC cost. We believe that our approach is a game-changing proposal towards a new generation of systems that combine the power of humans and machines.

REFERENCES

[1] O. Alonso and R. Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. In *ECIR*, 2011. Springer-Verlag.

[2] Amazon mechanical turk. www.mturk.com.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC*, 2007.

[4] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[5] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *SIGIR*, 1999.

[6] P. Bohannon, N. Dalvi, Y. Filmus, N. Jacoby, S. Keerthi, and A. Kirpal. Automatic web-scale information extraction. In *SIGMOD*, 2012.

[7] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[8] D. T. Bollegala, Y. Matsuo, and M. Ishizuka. Relational duality: unsupervised extraction of semantic relations between entities on the web. In *WWW*, 2010.

[9] R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *HLT*, 2005.

[10] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAI*, 2010.

[11] Z. Ce, N. Feng, R. Christopher, and S. Jude. Big data versus the crowd: Looking for relationships in all the right places. In *ACL*, 2012.

[12] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. R. Reiss, and S. Vaithyanathan. Systemt: an algebraic approach to declarative information extraction. In *ACL*, 2010.

[13] Crowdflower: Crowdsourcing, labor on demand. http://crowdflower.com.

[14] M.-C. de Marneffe, B. MacCartney, and C. D. Manning. Generating typed dependency parses from phrase structure trees. In *LREC*, 2006.

[15] J. Deng, W. Dong, R. Socher, L. jia Li, K. Li, and L. Fei-fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[16] P. DeRose, X. Chai, B. J. Gao, W. Shen, A. Doan, P. Bohannon, and X. Zhu. Building community wikipedias: A machine-human partnership approach. In *ICDE*, 2008.

[17] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 2011.

[18] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 2005.

[19] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[20] Y. Fang and K. C.-C. Chang. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In *WSDM*, 2011.

[21] C. Fellbaum, editor. *WordNet An Electronic Lexical Database*. The MIT Press, Cambridge, MA ; London, May 1998.

[22] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In *SIGMOD*, 2011.

[23] T. Furche, G. Gottlob, G. Grasso, O. Gunes, X. Guo, A. Kravchenko, G. Orsi, C. Schallhart, A. Sellers, and C. Wang. Diadem: domain-centric, intelligent, automated data extraction methodology. In *WWW*, 2012.

[24] R. Gupta and S. Sarawagi. Joint training for open-domain extraction on the web: exploiting overlap when supervision is limited. In *WSDM*, 2011.

[25] P. G. Ipeirotis and P. K. Paritosh. Managing crowdsourced human computation: a tutorial. In *WWW*, 2011.

[26] D. Karampinas and P. Triantafillou. Crowdsourcing taxonomies. In *ESWC*, 2012.

[27] Z. Kozareva and E. Hovy. Learning arguments and supertypes of semantic relations using recursive patterns. In *ACL*, 2010.

[28] E. Law and L. von Ahn. *Human Computation*. Morgan & Claypool Publishers, 2011.

[29] A. Marcus, E. Wu, D. Karger, S. Madden, and R. Miller. Human-powered sorts and joins. *Proc. VLDB Endow.*, 2011.

[30] A. Marcus, E. Wu, S. Madden, and R. C. Miller. Crowdsourced databases: Query processing with people. In *CIDR*, 2011.

[31] Mausam, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni. Open language learning for information extraction. In *EMNLP-CoNLL*, 2012.

[32] N. Nakashole, G. Weikum, and F. Suchanek. Patty: a taxonomy of relational patterns with semantic types. In *EMNLP-CoNLL*, 2012.

[33] R. Navigli and S. P. Ponzetto. Babelnet: building a very large multilingual semantic network. In *ACL*, 2010.

[34] F. Niu, C. Zhang, C. Re, and J. W. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. In *VLDS*, 2012.

[35] M. Paşca. Ranking class labels using query sessions. In *HLT*, 2011. Association for Computational Linguistics.

[36] A. G. Parameswaran, H. Park, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: declarative crowdsourcing. In *CIKM*, 2012.

[37] A. G. Parameswaran and N. Polyzotis. Answering queries using humans, algorithms and databases. In *CIDR*, 2011.

[38] S. Riedel, L. Yao, and A. McCallum. Modeling relations and their mentions without labeled text. In *ECML PKDD*,2010.

[39] C. Sarasua, E. Simperl, and N. F. Noy. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *ISWC*, 2012.

[40] P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. Open mind common sense: Knowledge acquisition from the general public. In *CoopIS/DOA/ODBASE*, 2002.

[41] R. Speer and C. Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, 2012.

[42] R. Speer, C. Havasi, and H. Surana. Using verbosity: Common sense data from games with a purpose. In H. W. Guesgen and R. C. Murray, editors, *FLAIRS Conference*. AAAI Press, 2010.

[43] F. M. Suchanek, G. Ifrim, and G. Weikum. Combining linguistic and statistical analysis to extract relations from web documents. In *KDD*, 2006.

[44] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.

[45] F. M. Suchanek, M. Sozio, and G. Weikum. Sofie: a self-organizing framework for information extraction. In *WWW*, 2009.

[46] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *Proc. VLDB Endow.*, 2011.

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

[48] L. von Ahn and L. Dabbish. Designing games with a purpose. *Commun. ACM*, 2008.

[49] L. von Ahn, M. Kedia, and M. Blum. Verbosity: a game for collecting common-sense facts. In *CHI*, 2006.

[50] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: crowdsourcing entity resolution. *Proc. VLDB Endow.*, July 2012.

[51] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.

[52] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *MobiSys*, 2010.

[53] C. Zhai. *Statistical Language Models for Information Retrieval*. Morgan & Claypool Publishers, 2008.

[54] J. Zhu, Z. Nie, X. Liu, B. Zhang, and J.-R. Wen. Statsnowball: a statistical approach to extracting entity relationships. In *WWW*, 2009.

[55] S. Kondreddi, P. Triantafillou, G. Weikum. HIGGINS: knowledge acquisition meets the crowds (Poster). WWW 2013 (Companion Volume).