

# Mixed-Initiative, Entity-Centric Data Aggregation using Assistopedia\*

Matthew Michelson, Sofus Macskassy and Steve Minton

Fetch Technologies  
841 Apollo St., Ste. 400  
El Segundo, CA 90245  
{mmichelson,sofmac,sminton}@fetch.com

## Abstract

Wikis allow for collaborators to collect information about entities. In turn, such entity information can be used for AI tasks, such as information extraction. However, these collaborators are almost exclusively human users. Allowing arbitrary software agents to act as collaborators can greatly enrich a wiki since agents can contribute structured data to complement the human-contributed, unstructured-data. For instance, agents can import huge volumes of structured data about entities, enriching the pages, and agents can update wiki pages to reflect real-time information changes (e.g., win-loss records in sports). This paper describes an approach that allows for both arbitrary software agents and human users to collaborate. In particular, we address three key problems: agents updating the correct wiki pages, policies for agent updates, and sharing the schema across collaborators. Using our approach, we describe creating entity-focused wikis which include the ability to create *dynamic* categories of entities based on their wiki pages. These categories dynamically update their membership based upon real-world changes.

## Introduction

Wikis, such as Wikipedia, essentially function as entity-centric data-integration frameworks where human users collectively aggregate different pieces of information about entities and publish the pieces either as text or structured data (in infobox tables). Increasingly, such collaborative collections of data are being leveraged for AI tasks, such as information extraction (e.g., (Fader, Soderland, and Etzioni 2009; Wu, Hoffmann, and Weld 2008; Lin, Etzioni, and Fogarty 2009)).

However, the collaborators for wikis are almost exclusively human users. This human exclusivity limits the capacity for importing data into wikis at scale, quickly, and with minimal cost. Further, human collaborators are limited in their ability track frequently changing real-world data and

update the wiki accordingly. In this paper we investigate relaxing this constraint such that arbitrary software agents also function as collaborators, enriching the wiki in a number of ways.

First, agents can automatically import huge volumes of structured data about entities. A user might want to add a new section to her wiki about NBA basketball. In this case, she could build an agent that extracts all player names, teams, positions, etc. from NBA.com and imports this data directly into the wiki as structured data about NBA player entities. This allows for quickly populating wikis at scale, with minimal human effort.

Second, agents can integrate data from various sources. For instance, our human user may identify webpages with players, their teams, and their coaches, another with players, their teams, and all of their statistics such as shooting percentage. Ideally she wants each of the NBA players' wiki page to have the player's name, team, coach and the full set of statistics. In this scenario, the first agent populates the wiki creating an NBA player page with structured attributes such as the name, team, and coach. The second agent then finds the appropriate pages on the wiki to update, and appends the statistical information. This way the human user did not have to perform the integration herself.

Finally, some pieces of information, such as each player's statistics, change frequently over time (in this case, almost everyday during the NBA season). Therefore, an agent can monitor the source with the players' statistics, and as the data changes, the agent updates the wiki accordingly. Therefore, software agents as collaborators contribute in two ways. First, they enrich the wiki by providing dynamically-updated structured-information that can be integrated from multiple sources. Second, they address tedious tasks on behalf of the human collaborators.

This paper presents Assistopedia, a framework that allows humans and arbitrary software agents to collaborate on building entity-centric wikis (wikis about entities, with attributes). These agents provide their own data from arbitrary sources (e.g., some agents extract data from webpages, some pull data from spreadsheets, etc.), and publish their records as structured data into the wiki creating tables on the wiki pages, which complement the edits of human users (e.g., human users can add both unstructured text to a page, or edit the tables published by the agents).

Our approach addresses three key problems that arise

\*This effort was in part sponsored by the Defense Advanced Research Projects Agency under grant number W31P4Q-09-C-0579. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense. Approved for Public Release, Distribution Unlimited  
Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

when agents act as collaborators in the wiki.

1. We describe how to ensure that agents update the correct pages with the structured data they provide. We model this an entity resolution problem where an agent must decide which (if any) entity (wiki page) to update. It becomes an entity resolution problem as different agents provide different attribute values for the same entity (e.g., “Matt Michelson” of “Fetch” versus “Matthew Michelson” of “Fetch Technologies.”)
2. We describe how agents can agree upon a shared vocabulary describing the agents’ attributes (which is necessary for resolution). We define “schema propagation,” a mechanism that loosely enforces a shared vocabulary describing the entities that grows over time. Schema propagation is a first-come, first-served mechanism where each agent contributes new attributes to a schema that are then propagated out to all entities (of the same type) in the wiki.
3. We address how to ensure agents behave well with the human collaborators. Our approach ensures this via publication policies that can be enforced per wiki and define the set of mixed-initiative policies that the agents follow regarding contributions. Further, our approach requires that all agents provide citations to sources from where they pull the data to build trust with the human users.

Finally, since the agents provide dynamically updated, structured data from multiple sources, we then describe how our framework allows for deep, structured search over the dynamically changing entities in the wiki.

### Assistopedia Overview

Assistopedia is a wiki framework that supports arbitrary software agents and human users as collaborators for building wikis about entities. We define an entity as the set of structured records that represent its attributes (appended with descriptive, unstructured text). This definition lets us break the collaborators into two classes: those that provide structured data for the wiki (agents) and those that provide both structured and unstructured data for the wiki (humans). Therefore, we assume that any agent that can create structured data can act as a contributor. Figure 1 shows the overall architecture for our Assistopedia framework.

As shown in the figure, our approach supports multiple agent types. Assistopedia allows any agent to participate as a collaborator as long as it produces structured data for its final output. As stated, we define entities as their set of attributes, which we model as a tuple {entity-id, attribute, attribute-value, source}. The role of the “Agent Agnostic Publisher,” then, is to take structured data produced by agents, and covert them into these tuples. These tuples are then stored in the “Entity-Based Record Store” (EBRS) which drives many of the capabilities supported by our approach. In particular, because the EBRS is a structured representation of the entities in the wiki, our approach leverages it for entity resolution, schema propagation, and for advanced, structured search over the entities. The EBRS is intimately tied with the wiki, such that all human edits to the wiki are instantly reflected in the EBRS. Therefore, both entity resolution and the advanced searching capabilities always access the freshest data. On the flip side, all changes to the EBRS (such as

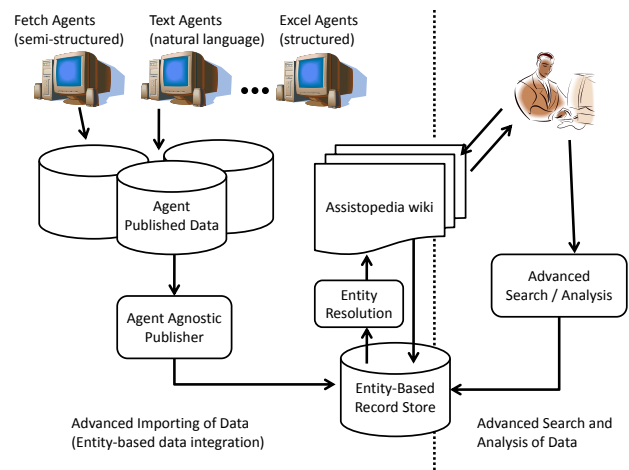


Figure 1: Assistopedia: the Mixed-Initiative wiki

those provided by an agent) are reflected on the wiki pages in real-time.

### Managing Pages with Entity Resolution

As stated, we model entities via their tuples, which are records of {attribute, value} pairs. Each wiki page reflects this modeling in its “entity table.” To make this clear, Figure 2 shows example Assistopedia pages for two NBA basketball players, provided by an agent that extracted and then imported the data from NBA.com. For trust purposes, we require that each agent sources its data, which is shown in the figure as the footnotes after each attribute value, containing a reference the original source. Since all agents must reference their attribute values, multiple agents supplying the same value for the same attribute will result in multiple references for that value (described later). This aids a human user’s understanding of which values supplied by agents may be more credible (based on redundancy across sources). In this sense, multiple agents are cooperating to cite their sources in the correct collaborative manner. Note that the page about Kobe Bryant has a human edit (adding the text “He is the best in the NBA.”) to show mixed-initiative collaboration.

As each agent supplies data to the wiki, the data is translated into a set of tuples by the Publisher. The entity resolution component then compares this tuple representation of the entity to the current state of the entities from the EBRS (which contains the most up-to-date entity tuples). Resolution then decides if the new tuples match an entity currently in the wiki, in which case the matching entity is updated with new data from the tuples, or if the tuples represent a completely new entity to be imported. Since agents may supply conflicting information (such as differing player heights) or represent the same attribute values differently (e.g., “Bob” is the same as “Robert”), entity resolution must handle such data inconsistencies when reasoning about entity similarity. Once resolved, if the tuples update an entity, it is the “publishing policy” (see subsection below) that dictates which attribute values are displayed on the wiki page. For example, a policy might always display the most re-

127.0.0.1 talk for this ip log in / creat

page discussion edit history

## Entity 280674026

Attribute	Value
FirstName	Kobe <sup>[1]</sup>
LastName	Bryant <sup>[1]</sup>
City	Los Angeles <sup>[1]</sup>
Team	Lakers <sup>[1]</sup>
School	Lower Merion HS <sup>[1]</sup>

Footnotes keep track of agent's source for values

1. ↑ 1.0 1.1 1.2 1.3 1.4 <http://www.nba.com>

He is the best in the NBA.

Log in / create account

page discussion edit history

## Entity 434315496

Attribute	Value
FirstName	LeBron <sup>[1]</sup>
LastName	James <sup>[1]</sup>
City	Cleveland <sup>[1]</sup>
Team	Cavaliers <sup>[1]</sup>
School	St. Vincent-St. Mary HS <sup>[1]</sup>

1. ↑ 1.0 1.1 1.2 1.3 1.4 <http://www.nba.com>

Figure 2: Two example Assistopedia pages

cently provided attribute value (which may conflict with previous values). We have designed Assistopedia to be flexible regarding which system is used for entity resolution (e.g., (Minton et al. 2005; Tejada, Knoblock, and Minton 2001; Christen 2008)), though unsupervised approaches are preferred (e.g., (Bhattacharya and Getoor 2006; Benjelloun et al. 2009)). In this way, new attributes or updated attributes supplied by an agent can be assigned to the correct entity in the wiki.

As described above, even when human users update the wiki, the state of the tuples in the EBRs is updated such that Assistopedia always has access to the latest information represented in the wiki for tasks such as entity resolution.

Assistopedia

Attribute	Value
School	Lower Merion HS <sup>[1][2]</sup>
FirstName	Kobe <sup>[1][2]</sup>
LastName	Bryant <sup>[1][2]</sup>
City	Los Angeles <sup>[1][2]</sup>
Team	Lakers <sup>[1][2]</sup>
Height	6-6 <sup>[2]</sup>
Weight	205 <sup>[2]</sup>
Age	31 <sup>[2]</sup>

Newly added from source #2

1. ↑ 1.0 1.1 1.2 1.3 1.4 <http://www.nba.com>

2. ↑ 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 <http://www.lakers.com/roster>

He is the best in the NBA.

Figure 3: An update to the wiki page by a software agent

Figure 3 shows the entity resolution component updating the entity representing Kobe Bryant. In this case, another agent, using information from the Lakers team website,<sup>1</sup> supplied extra attributes such as the “height,” “weight,” and “age” to the existing entity. Because it also supplied

<sup>1</sup>[www.lakers.com/roster](http://www.lakers.com/roster)

the name, team, etc., the system decided that these new attributes should update the existing Kobe Bryant entity. Note that the references in the entity table are updated, showing that the confirmed values are now supported by two sources, both [lakers.com](http://www.lakers.com) and [nba.com](http://www.nba.com). Lastly, the human contributed edit (the sentence at the bottom) was not changed, even though the page was updated by an agent. This figure shows a simple example of full mixed-initiative collaboration through entity resolution. Two agents representing two disparate sources combined their information with a human edit for this entity.

### Schema Propagation

When defining entities by their attributes, while allowing the open addition/updating of these entities, it becomes very important to keep a consistent schema (set of attributes) that define the entity. One way to do this is to enumerate all possible attributes for an entity, a priori, and have all contributors (both agents and humans) agree to the set. An example would be to build an ontology of NBA players and conform to it that such that any attribute in a record for an entity must come from the ontology. However, as Wikipedia and other social media have shown, often times an agreed upon set of attributes can develop organically from users of the system, without the daunting task of both creating the ontology and having all users agree to it. This organically grown schema is called a “folksonomy.” In this work, we investigate a mechanism that allows for a folksonomy to develop (which is much more flexible than defining an ontology a priori), but that also aids in enforcing adherence to this folksonomy.

We call our approach “schema propagation.” Schema propagation is a first-come, first-served approach to folksonomy development. In particular, the first collaborator (agent or human) to add an attribute chooses a name for that attribute, and then all current entities and subsequent new entities then have this attribute propagated to them upon any change to an entity table. Schema propagation is based on type, so attributes are only propagated amongst entities that share an entity type (e.g., NBA players). As an example,

consider Figure 4, where the “height,” “weight,” and “age” attributes that were newly added to the entity in Figure 3 have been propagated to the entity of Figure 4 (with empty values). In this way, all current and subsequent NBA player entities now have a notion of the “weight,” “height,” and “age” attributes.

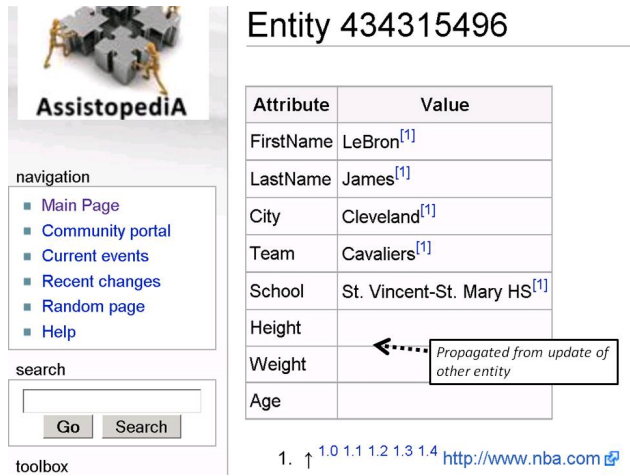


Figure 4: Schema propagation to this entity

This implies the assumption that when agents supply records to the Agent Agnostic Publisher, they perform their own schema mapping (in the future, we plan to investigate incorporating automatic schema mapping into the Agent Agnostic Publisher). To do the mapping, agents can first request the latest attribute set (folksonomy) from Assistopedia’s EBRS, and then map their structured representation to this folksonomy. This attribute set is exactly that one built via schema propagation. Therefore, all incoming agent schema are mapped to the latest Assistopedia folksonomy.

Schema propagation is also important for human users collaborating with the agents. To support this, if a human user adds his or her own record, and only specifies part of the schema as attributes in the table, the Assistopedia fills in the rest of the attributes (according to the folksonomy) to guide future schema consistency. This is shown in the following figures. In Figure 5, we see the preview page showing an entity table that a human user has designed for the NBA player Michael Jordan.

Once the user submits this page to Assistopedia, the rest of the attributes are filled in, as shown by Figure 6. Similarly to agent imports/updates, if a human user adds an attribute that is new to the Assistopedia folksonomy, it will be propagated to all other entities, just as if an agent had added a new attribute. This schema-propagation design allows for both the flexibility of defining the folksonomy on demand, by all collaborators, while urging the collaborators to conform to the folksonomy. We note that refactoring and managing this folksonomy is an area of future work.

### Agent Publishing Policies

We require that all agents abide by two contracts. First, any agent that wishes to publish into Assistopedia produces tuples of its data (rows of attribute, value pairs), regardless of

whether it extracts data from text, spreadsheets, queries of XML, etc. Second, we require that each agent maps its own attribute names to the current set of attribute names in Assistopedia (e.g., Assistopedia does not yet provide schema mapping capabilities).

However, Assistopedia is mixed-initiative, such that beyond the usual wiki functions of freely editing text around entity tables, human users can both edit the tables written by agents, and create totally new entity tables which will be incorporated into Assistopedia as new entities. In the spirit of mixed-initiative collaboration, any text not in an entity table is guaranteed to never be changed by an agent, so agents can be trusted to only update agent-designated areas on the page. However, allowing both human users and agents to add and update the entity tables gives rise to the necessity of “publishing policies.” A publishing policy reflects how information from different collaborators should be given precedence, which is generally reached by consensus in the community. For our approach, we identified a few initial policies, described in Table 1.<sup>2</sup> This is really a third contract on the agents. While they do not explicitly agree to the contract (because they don’t control their own updates, Assistopedia does) there is an implied contract on how their data will make it into the wiki based upon the policy.

Table 1: Different policies for mixed-initiative wiki updates

<p><b>Freshness Policy: The freshness of data takes precedence</b></p> <p><b>Requirements</b></p> <ol style="list-style-type: none"> <li>1. Up-to-dateness of data is essential</li> <li>2. Any data is better than no data</li> </ol> <p><b>Policy</b></p> <ol style="list-style-type: none"> <li>1. First-come, first-served on data value updates</li> <li>2. Anyone (agent or human) can overwrite anyone else’s value</li> <li>3. No blanks for data (better to have some data value than none)</li> <li>4. You may only change a value if it’s different from a previous value you set</li> </ol>
<p><b>Consensus Policy: Only agreed upon values take precedence</b></p> <p><b>Requirements</b></p> <ol style="list-style-type: none"> <li>1. Data must have agreed upon value</li> <li>2. No data value is better than a mysterious data value</li> </ol> <p><b>Policy</b></p> <ol style="list-style-type: none"> <li>1. Data values update only when more than X collaborators agree on value</li> <li>2. If two values have more than X agreements, value is removed until consensus is reached amongst factions</li> <li>3. You may only vote for a value once</li> </ol>
<p><b>Human Policy: The human provided data always takes precedence</b></p> <p><b>Requirements</b></p> <ol style="list-style-type: none"> <li>1. Human values always supercede agent supplied values</li> </ol> <p><b>Policy</b></p> <ol style="list-style-type: none"> <li>1. An agent may never overwrite a value provided by a human user</li> <li>2. Humans may overwrite each other according to their own policy</li> <li>3. An agent may only overwrite another agent or a blank value</li> </ol>

### Dynamic Categories

The above subsections describe how our approach handles three key problems when agents act as collaborators in the wiki. Namely, how they update the correct pages, how they agree on a shared vocabulary, and how they “play fairly” with the other collaborators. Now, we describe one of the benefits of having agent collaborators. Agents can monitor sources for changes and update the wiki accordingly, without putting the onus on the human collaborators. This leads to a powerful capability: “Dynamic Categories.”

In Wikipedia, a category is a clustering of pages that meet some criteria. Each page in the cluster is tagged with the name of the category, and users can go to the category’s page to see a listing of all members (along with optional text describing the category). For instance, Figure 7 shows the Wikipedia category of “Computer Companies,” which

<sup>2</sup> Assistopedia currently runs using the “Freshness Policy.”

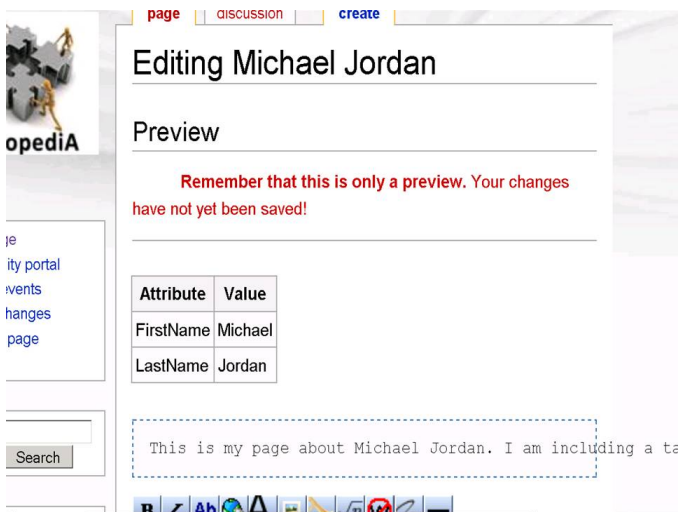


Figure 5: A human edited wiki page

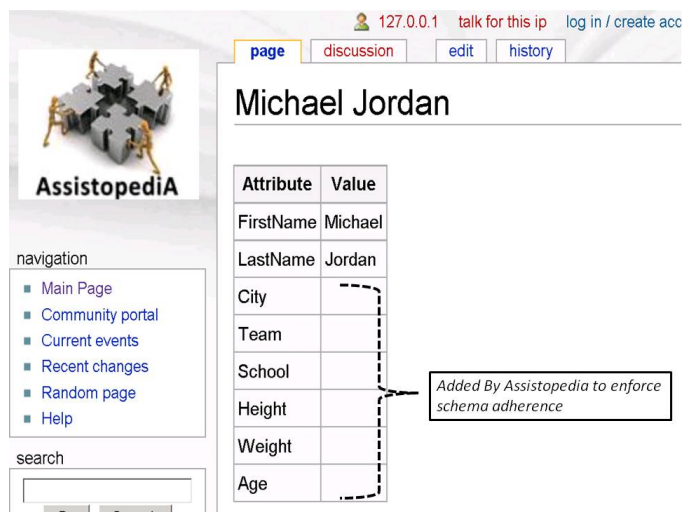


Figure 6: Page from Figure 5, after schema propagation

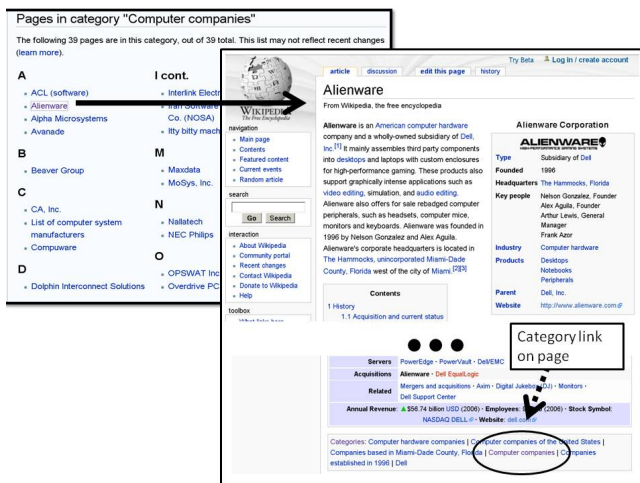


Figure 7: Traditional Category on Wikipedia

lists links to wiki pages of the cluster of computer companies. The figure also shows that by following this link to the wiki page, the page also has a link back to the categories to which it belongs shown at the bottom of the page. Human users generally create these categories by adding such links to the bottom of the pages, which can be a tedious and error prone task, especially if the cluster changes frequently over time.

In the context of Assistopedia, a dynamic category capability allows users to define clusters by building a structured query over the records, such as all NBA players who play for the “Lakers.” Then, all entities that meet this criteria, which are those that would be returned by the query (select where team equals Lakers), are put into the category. The unique feature of our approach to categories, which differentiates it from traditional wiki categories, is that the clustering (and reclustering) is automatic. In traditional wikis, users manually add and update the members of the categories. How-

ever, our categories are defined by structured queries over the records stored in the EBRS, and therefore the cluster can be built automatically. Further, and most importantly, since the EBRS changes whenever collaborators add entities or change the records of entities, the clusters will then change dynamically as well, since all (re)clustering is based upon the EBRS. This is especially useful for categories that change over time, since an agent can be tasked with monitoring a source for changes, and updating Assistopedia to reflect the changes, which are then reflected in the user’s desired category.

This process is shown in Figures 8 and 9. Starting in the top left of Figure 8, we see that a user has used the form<sup>3</sup> to build a category called “Lakers Players.” For this category, any entity whose team is Lakers will be a member of this category. The next box, in the middle of the figure, shows that this category is generated and treated as any other category using the style of Wikipedia. The bottom of this figure shows the two links to the correct players’ pages,<sup>4</sup> demonstrating both that the correct entities are clustered and that each entity’s page was updated to include a pointer to this new category for which they are members.

Our dynamic categories are treated as any other category, making them familiar to users of Wikipedia, but they differ because as new entities are added to the wiki or previous entities are updated, the membership of these categories changes dynamically. The left side of Figure 9 shows a human collaborator updated the city and team to different values,<sup>5</sup> and upon saving her edits, the category was automatically removed from the page. The right side of Figure 9 shows that the membership of the category was also updated, reflecting the removal of this player.

<sup>3</sup>The form dynamically creates the set of attributes to use for building queries based upon the schema that develops over time via propagation.

<sup>4</sup>We only include two Lakers players for clarity.

<sup>5</sup>Since it was a human edit, the references are discarded

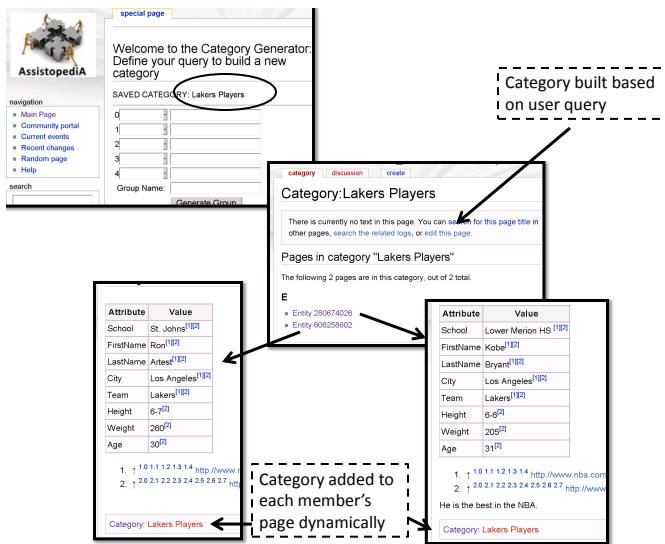


Figure 8: Building Dynamic Categories

### Related Work

The use of Wikis for AI tasks has grown tremendously (hence the need for this workshop). In our case, we use wikis as a mixed-initiative integration framework that requires using entity resolution. In our research we leverage an unsupervised method for resolving entities based on their records, but previous work has demonstrated success relating records in relational databases to text either on Wikipedia (Downey, Ahuja, and Anderson 2009) or in general text collections (Chakaravarthy et al. 2006). This is certainly a direction that complements our approach, where we could use such methods to improve the overall entity resolution in our framework.

Also, previous work demonstrated the ability to augment the infobox tables (structured representation) of entities on Wikipedia using information extraction methods (Wu, Hoffmann, and Weld 2008). Again, this is an interesting complementary technology, as Assistopedia is agnostic to the agents that produce records, such that this could become another agent for publishing data into the system.

### Conclusions

In this paper we presented Assistopedia, a framework that allows both arbitrary software agents and human users to collaborate in aggregating information about entities in a Wiki. Our approach focuses on breaking down agent supplied information into structured records, which can then be used for entity resolution on agent updates, and for searching the wiki in a structured format.

There is certainly a significant amount of research to accomplish our goal of truly mixed-initiative data integration using wikis. First, we would like to incorporate schema matching into Assistopedia's Agent Agnostic Publisher, such that the onus falls less on the agents, so they could more easily publish into the system. Second, we would like to leverage previous work on extracting structured data from various unstructured sources such as Wikipedia (e.g., (Wu, Hoffmann, and Weld 2008)), lists on the Web (e.g., (Gupta and Sarawagi 2009)), and free Web text (e.g., (Banko et al. 2007)). By incorporating many types of agents we can open

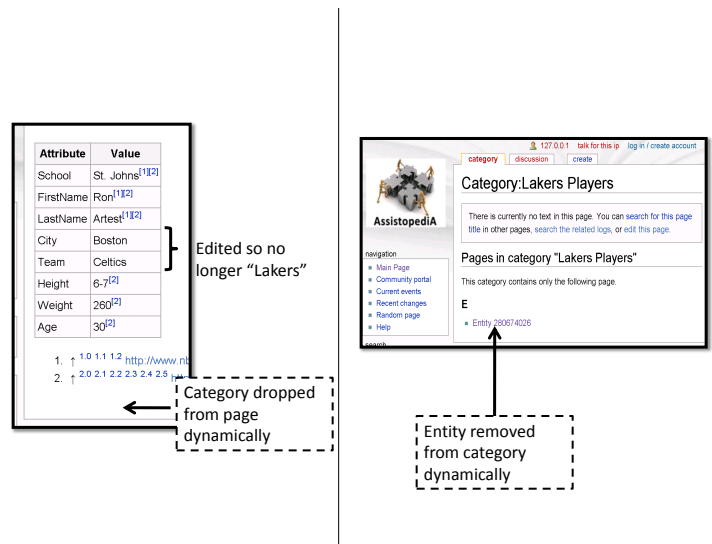


Figure 9: Automatic Updates to the Dynamic Category

up the possibilities of sources to aggregate within Assistopedia. Finally, we will investigate methods of linking structured records to unstructured text (Downey, Ahuja, and Anderson 2009; Chakaravarthy et al. 2006) to improve the entity resolution of our method.

### References

Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M.; and Etzioni, O. 2007. Open information extraction from the web. In *Proc. of IJCAI*.

Benjelloun, O.; Garcia-Molina, H.; Menestrina, D.; Su, Q.; Whang, S. E.; and Widom, J. 2009. Swoosh: a generic approach to entity resolution. *VLDB J.* 18(1):255–276.

Bhattacharya, I., and Getoor, L. 2006. A latent dirichlet model for unsupervised entity resolution. In *Proc. of SDM*.

Chakaravarthy, V. T.; Gupta, H.; Roy, P.; and Mohania, M. 2006. Efficiently linking text documents with relevant structured information. In *Proc. of VLDB*, 667–678.

Christen, P. 2008. Febrl -: an open source data cleaning, deduplication and record linkage system with a graphical user interface. In *Proc. of KDD*, 1065–1068.

Downey, D.; Ahuja, A.; and Anderson, M. 2009. Learning to integrate relational databases with wikipedia. In *Proc. of WikiAI*.

Fader, A.; Soderland, S.; and Etzioni, O. 2009. Scaling wikipedia-based named entity disambiguation to arbitrary web text. In *Proc. of WikiAI*.

Gupta, R., and Sarawagi, S. 2009. Answering table augmentation queries from unstructured lists on the web. *Proc. VLDB Endow.* 2(1):289–300.

Lin, T.; Etzioni, O.; and Fogarty, J. 2009. Filtering information extraction via user-contributed knowledge. In *Proc. of WikiAI*.

Minton, S. N.; Nanjo, C.; Knoblock, C. A.; Michalowski, M.; and Michelson, M. 2005. A heterogeneous field matching method for record linkage. In *Proceedings of ICDM*, 314–321.

Tejada, S.; Knoblock, C. A.; and Minton, S. 2001. Learning object identification rules for information integration. *Information Systems* 26(8).

Wu, F.; Hoffmann, R.; and Weld, D. S. 2008. Information extraction from wikipedia: Moving down the long tail. In *In Proc. of SIGKDD*.