A Multi-Agent System with Reinforcement Learning Agents for Biomedical Text Mining

Michael Camara
Allegheny College
Department of Computer Science
520 North Main Street
Meadville, PA
camaram@allegheny.edu

Oliver Bonham-Carter
University of Nebraska at Omaha
College of Information Science and Technology
6001 Dodge Street
Omaha, NE
obonhamcarter@unomaha.edu

Janyl Jumadinova
Allegheny College
Department of Computer Science
520 North Main Street
Meadville, PA
jjumadinova@allegheny.edu

ABSTRACT
Due to the expanding growth of information in the biomedical literature and biomedical databases, researchers and practitioners in the biomedical field require efficient methods of handling and extracting useful information. We present a novel framework for biomedical text mining based on a learning multi-agent system. Our distributed system comprises of several software agents, where each agent uses a reinforcement learning method to update the sentiment of a relevant text from a particular set of research articles related to specific keywords. Our system was tested on the biomedical research articles from PubMed, where the goal of each agent is to accrue utility by correctly determining the relevant information that is communicated with other agents. Our results tested on the abstracts collected from PubMed related to muscular atrophy, Alzheimer’s disease, and diabetes show that our system is able to appropriately learn the sentiment score related to specific keywords by parallel and distributed analysis of the documents by multiple software agents.

Categories and Subject Descriptors
J.3 [Applied Computing]: Life and medical sciences

General Terms
Design, Experimentation

Keywords
Multi-agent, reinforcement learning, biomedical text mining

1. INTRODUCTION
With the exponential growth of scientific information over the last several centuries [21], the volume of published biomedical research has been expanding at a particularly increasing rate. As of 2015, the PubMed database contains over 24 million records, and between 1986 and 2010 the total number of citations in PubMed has been growing annually at a 4% rate [11]. Due to this explosive growth, it is very challenging to keep up to date with all of the new research discoveries and methods within biomedical research. Poor communication creates a disconnection between highly specialized fields and subfields [5], and it is a challenge to share the wealth of knowledge from discovery between the research disciplines. With the current increasing rate of growth in published biomedical research, it is very likely that important connections between individual elements of biomedical knowledge, that could lead toward practical use in the forms of diagnosis, prevention and treatment, are not being created.

Text mining is an automated approach to harvesting knowledge from a wide and diverse corpus of records. Text mining and information extraction have been gaining popularity as a way to aid researchers with a large volume of information [12]. Biomedical text mining allows researchers to identify needed information more efficiently and discover relationships obscured by the large volume of available information that exists in the literature. Information extraction and text data analysis can be particularly relevant and useful in biomedical research, in which current information about complex processes involving genes, proteins and phenotypes is crucial.

Various data mining tools and methods have been recently established to help researchers and physicians in making more efficient use of the existing research [11, 15]. Most of the existing tools operate as a centralized data mining process and do not incorporate learned information during this process. However, it may be necessary to collect and analyze different distributed sources of voluminous data, potentially stored across different databases. Distributed data mining (DDM) enables mining data sources regardless of their source [19]. Many distributed data mining systems adopt the multi-agent system architecture [6, 8]. A multi-agent system is a collection of intelligent software agents that interact with each other. An intelligent agent is an autonomous entity or software program that automatically performs tasks on behalf of the user to accomplish some goal. A multi-agent system is able to achieve an efficient, flexible,
and adaptable system that allows for better communication, interaction and management among different components of a distributed data mining system [3].

In this paper, we present a multi-agent based framework with learning that processes PubMed research article abstracts in a distributed way. Our system allows for the integration of the learned information from the extracted data by the individual agents throughout the information extraction process. Software agents within our system utilize a reinforcement learning algorithm, which provides a reliable learning solution, despite the lack of existing labeled data used in our experiments. We tested our framework on the PubMed research articles related to muscular atrophy, Alzheimer’s disease, and diabetes. In our experiments, the software agents in our system were tasked with learning the sentiment information as they processed and analyzed each abstract from the assigned data set. Abstracts are short relevant texts to describe a larger body of work. Since they must be specific, the knowledge we extract from them is likely to be factual and non-ambiguous. Our experiments are based on this factual corpus. Our results demonstrate that the proposed multi-agent text mining framework is able to appropriately learn the sentiment score related to specific keywords by parallel and distributed analysis of the documents by multiple software agents. In summary, the contributions of this paper are as follows:

1. A multi-agent system for distributed text mining of biomedical information.
2. An application of the reinforcement learning algorithm by each software agent to correlate information.
3. The development of a novel scientific workflow involving data extraction and processing of the data.
4. An experimental study which proves the concept of our idea: an automated text mining framework of data analysis, driven by a multi-agent system.

2. RELATED WORK

Text mining is becoming a more prominent research tool to cope with the increasing availability of large data. Previous text mining tools have concentrated on extracting information for human consumption (text summarization, document retrieval), assessing document similarity (document clustering, key-phrase identification), and extracting structured information (entity extraction, information extraction) [20, 21].

Important applications for text mining during the recent years have been in biomedical research due to its data which is now more readily available. Research in text mining for biomedical use has concentrated on several different aspects. Some recent studies have concentrated on named entity recognition that can identify, within a collection of text, all of the instances of a specific name [5, 23]. Other researchers developed techniques for text classification that attempt to automatically determine whether a document, or part of a document, has particular characteristics of interest [10]. Other types of tools were created for relationship extraction that detect occurrences of a specified type of relationship between a pair of entities of given types [22].

There are a number of articles that survey the existing search tools for biomedical literature to elucidate the differences between search tools. For example, Kim et al. [9] examine the underlying steps and categorization of several biomedical literature search tools, and give an overview of the input and output formats used in these tools. The authors then classify the tools by their behavior patterns, namely: starting, chaining, browsing, differentiating, monitoring and extracting. They conclude that although different tools perform better in these different facets, there is a need to develop better solutions for differentiating documents. In another review article, Lu [11] examines 28 unique web tools for searching the biomedical literature, similar to the National Center for Biotechnology Information’s (NCBI) PubMed web service. All of the reviewed tools search through a large collection of biomedical literature based on a user specified query, returning selected documents that, ideally, closely match the user’s request. Each of the selected tools was placed into one of four categories based on their most notable features: ranking search results, clustering results into topics, extracting and displaying semantics and relations, or improving the search interface and retrieval experience. After analyzing each in turn, Lu highlights some key features; for instance, the usefulness of document clustering to group together similar documents using biomedical controlled vocabularies/ontologies as representative topic terms.

Recently, some researchers used a multi-agent system for text mining in order to create a distributed data mining system [3, 8]. Di Fatta and Fortino [7] propose a multi-agent system framework for distributed data mining based on a peer-to-peer model. They highlight the importance of such distributed systems due to the overwhelming increase in the amount of data and high performance costs. The authors use their model for data collection on molecular structures, and based on their results they claim that the framework is further suitable for large-scale, heterogeneous computational environments such as grids. Chaimontree et al. [4] describe a generic multi-agent data mining (MADM) framework that is able to handle large amounts of data, provides adequate security, and yields the best clusters possible through multi-agent interactions. Their framework uses JADE (Java Agent Development Environment) along with five different types of agents, each with their own distinct purpose. By comparing the F-Measure of several published clustering algorithms for given data sets, the best clustering algorithm is identified. In [18], Tong et al. describe a data mining multi-agent system (DMMAS) to model chronic disease data, with type II diabetes as their primary study domain. They focus on improving the efficiency of data mining using parallel and distributed data mining algorithms. While Chao and Wong [6] propose a multi-agent data mining workbench to assist physicians in making objective and reliable diagnoses. Their proposed multi-agent learning paradigm, called DiMAS, makes use of heterogeneous intelligent data mining agents, each being responsible for a specific and independent task.

In this paper, we present a learning multi-agent system for mining biomedical research articles. In contrast to the previous works that utilize multi-agent system framework for data mining, we consider exploring and finding relations between the text in biomedical research articles. Another contribu-
tion our paper makes is an application of the reinforcement learning by the individual agents to learn appropriate relations better through sentiment. Finally, we present a completely automated workflow, starting with data collection and storage, and finishing with the analysis.

3. TEXTMED: A LEARNING MULTI-AGENT SYSTEM

3.1 Data Collection and Preprocessing

In order to collect the data necessary for our multi-agent system, we first created a framework called Lister, which automatically downloads and decompresses every article available through the PubMed database [14] into NXML format. Lister then parses all of the available articles with a given keyword using a bag of words approach, producing the abstract of each article containing that keyword at least once. Further, all forms of punctuation were removed from these abstracts, in addition to irrelevant stop words that could potentially interfere with subsequent analysis. This process can be repeated indefinitely for any number of unique keywords, producing a subset of relevant abstracts after each execution.

At the end of the data collection, the data set $D$ obtained from Lister is divided into a subset of data sets $D_i$, so that each subset can be processed by one software agent $i$. Another aspect of preprocessing the data involves the creation of a list of keywords to isolate in the corpus. This list conforms to the MeSH (Medical Subject Headings) descriptors created and maintained by the U.S. National Library of Medicine, containing over 27,000 unique entries relevant specifically to the biomedical field in 2015 [13]. The software agents learn relevant sentiment as related to the secondary keyword obtained from the list of MeSH keywords and the primary keyword used by the Lister.

Figure 1 shows the general workflow for our multi-agent system. All of the results obtained after the analysis by our system are stored locally in a database for quick and convenient access.

3.2 Learning Intelligent Agents

Our multi-agent system, called TextMed, begins by creating separate agents for each subset of abstracts created by Lister. Figure 2 describes the workflow of one individual agent $i$. Since the agents in our system are homogeneous, the same process applies to all of the agents. Each agent starts by parsing its designated data set using the same list of keywords created previously. Each agent first determines whether a given keyword, or its pluralized form, appears in the abstract at least once. If the keyword does not appear in the abstract then the next keyword is tested; otherwise, the agent begins the sentiment analysis of each match using a variable that we call proximity. The proximity variable indicates the number of words to the left and to the right of a keyword match that will be included in the subsequent sentiment analysis.

Sentiment analysis is performed by a program called SentiStrength [17]. This program was originally designed to perform sentiment analysis on short social web texts, such as Twitter and YouTube comments; but we have leveraged its capabilities for use with biomedical texts [16]. In its simplest form, SentiStrength accepts a string of text as input and produces both positive and negative sentiment scores as output: the higher the magnitude of the score, the higher the positive or negative correlation of the text. By combining the positive and negative scores we create a composite score in the range of -4 (strong negativity) to 4 (strong positivity), with a score of 0 indicating neutral sentiment. If multiple matches of a keyword are found in a document, then sentiment analysis is performed on each match individually and then averaged together to generate the local calculated sentiment, $ls_{k,d}$ of the keyword $k$ in the document $d$ for the proximity value tested.

After agent $i$ processes each document and calculates the local sentiment for keyword $k$ in that document, it also updates the global sentiment value, $gs_k$, of keyword $k$ across all documents that all agents have processed so far. Initially, when a particular keyword is found for the first time in one of the documents, the corresponding keyword object records the local calculated sentiment as the new global sentiment for the given proximity value ($gs_k = ls_{k,d}$). However, if the keyword has been found by one of the agents in previous documents, then the agent uses a reinforcement learning algorithm to produce a new learned global sentiment.

Reinforcement learning has been frequently used and stud-
Figure 2: Individual Agent Workflow, demonstrated on agent $i$

ied in multi-agent systems [1, 2]. This learning paradigm relies on agents choosing an action and then receiving a scalar reward based on how the state of the environment changes. In our experiments, the state of the environment correlates to the global sentiment score for each keyword. By maintaining historical data on the reward given for each action performed, we can design our agents to seek the most optimal reward possible based on both short-term and long-term goals. This type of agent learning was chosen due to the lack of existing sentiment data for keywords tested in the scope of biomedical research. Without such reference data, we needed a way to determine the validity of each calculated sentiment score; and reinforcement learning provided a reliable solution to the problem. The reinforcement learning algorithm used by each agent is presented as Algorithm 1 and is described in more detail below.

Data: $l_{k,d}$, $g_{k}$
Result: $g_{k}$

Algorithm 1: Agent Reinforcement Learning Algorithm

In our system, the reward, $R_k$ for keyword $k$ after $N$ number of documents have been processed is calculated as the sum of the magnitude of the difference between the anticipated next state (global sentiment) and the local sentiment for each document divided by the total number of processed documents so far containing the keyword as shown in Equation 1.

$$R_k = \frac{\sum_{i=d}^{N} |g_{k} - l_{k,d}|}{N}$$  

(1)

Further, smaller numbers are favored more than large numbers since they show smaller variance with the calculated sentiment in previously parsed documents. During reinforcement learning, agent $i$ chooses one of 6 policies to maximize the potential reward for the agent. These policies are designed to apply a simple change to the local sentiment score; and reinforcement learning provided a reliable solution to the problem. The reinforcement learning algorithm used by each agent is presented as Algorithm 1 and is described in more detail below.

While the previous calculations attempt to maximize the short-term reward, we use a Q-learning reinforcement learning approach (a model-free reinforcement learning technique) to allow consideration for long-term reward as well. Each keyword object contains a Q-table: a two dimensional array representing historical rewards for each state-action pair, with each index initialized to the largest, least favorable reward possible. The learning algorithm iterates through all possible pairings in the Q-table given the current local sentiment, eventually finding the smallest, most favorable reward and the corresponding action needed to attain the indicated next state.

The reward values obtained from both of these approaches are compared, and the smaller, more favorable number is chosen as the optimal reward, and its corresponding action is marked as the optimal action. Finally, in order to address the exploration-exploitation trade-off, we introduce an infrequent random chance for the optimal action to increase or decrease by 1, allowing potentially better states to be reached and recorded. The next state is determined by summing the local sentiment with the optimal action, and this value is used to explicitly adjust the global learned sentiment for the keyword. The Q-table for the keyword is then updated following the formula in Equation 2. Note that
\[ Q[ls|gs] = (1 - \delta)Q[ls|gs] + \delta \cdot \text{(calculatedBestReward + \gamma \cdot \text{optimalReward})} \]  

Once an agent has finished parsing one of its documents, it calculates and stores the amount of utility it has earned from that document. Utility, \( U_{i,d} \), for agent \( i \) is calculated as the average reward given among all keywords found in a particular document as shown in Equation 3, with smaller utility values indicating smaller, more optimal rewards.

\[ U_{i,d} = \frac{\sum_{k=1}^{\text{numKeywords}} R_{k,d}}{\text{numKeywords}} \]  

### Table 1: Policies used in the reinforcement learning algorithm

<table>
<thead>
<tr>
<th>Policy #</th>
<th>Description of the global sentiment calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( gs_k = l_{s_k,d} ) if ( ls_{k,d} &lt; gs_k ), otherwise ( gs_k = l_{s_k,d} + 1 ). Otherwise, ( gs_k = l_{s_k,d} - 1 ).</td>
</tr>
<tr>
<td>1</td>
<td>( gs_k = l_{s_k,d} + 1 ) if ( ls_{k,d} &lt; gs_k ), otherwise ( gs_k = l_{s_k,d} - 2 ). Otherwise, ( gs_k = l_{s_k,d}/2 ).</td>
</tr>
<tr>
<td>2</td>
<td>( gs_k = l_{s_k,d} ) if ( gs_k &gt; 0 ), otherwise ( gs_k = l_{s_k,d}/2 ).</td>
</tr>
<tr>
<td>3</td>
<td>( gs_k = l_{s_k,d} ) if ( gs_k &lt; 0 ), otherwise ( gs_k = l_{s_k,d}/2 ).</td>
</tr>
<tr>
<td>4</td>
<td>Choose randomly to increase or decrease ( gs_k ) by 1.</td>
</tr>
</tbody>
</table>

\( \delta \in (0, 1) \) is a learning rate and \( \gamma \in (0, 1) \) is a discount factor, these values are arbitrary set to 0.5 by default in our experiments. By using the reward-based learning process each agent incorporates feedback from its own historical experience through the local sentiment and also from the other agents’ by considering the global sentiment when calculating the reward.

### 4. EXPERIMENTAL RESULTS

During our experimental analysis we analyzed the performance of various parameters within our system, such as the proximity parameter, number of agents, and we evaluated the results concerning the output of our reinforcement learning algorithm including the relationship between the sentiment values and the reward values. We produced heatmap graphs to display our results in addition to reward and utility graphs. A heatmap is a color-coded matrix of numerical values which have been clustered across the top and the side. The heatmaps convey three distinct parameters as represented by the \( x \)-axis, \( y \)-axis, and scale that ranges between blue and red, cooler and warmer colors, respectively. Heatmaps are useful in comparing several different large sets of data together in terms of their numeric information.

### Table 3: The data sets used in our experiments.

<table>
<thead>
<tr>
<th>Data Set Primary Keyword</th>
<th>Number of Abstracts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muscular Atrophy</td>
<td>400</td>
</tr>
<tr>
<td>Alzheimer’s</td>
<td>2700</td>
</tr>
<tr>
<td>Diabetes</td>
<td>10000</td>
</tr>
</tbody>
</table>

### 4.1 Data Sets

We conducted experiments on three data sets of varying sizes for 10, 50, and 100 agents, as shown in Table 3, consisting of research article abstracts that were obtained from the PubMed database. All three data sets were obtained by using our Lister tool, which first downloaded and decompressed all of the articles available through the PubMed database into NXML files. We used the PubMed data from June 2015. Lister then collected abstracts containing one of the keywords from the downloaded articles. We arbitrarily chose and used the keywords, **muscular atrophy**, **Alzheimer’s**, and **diabetes**, separately to obtain abstracts containing these words, which were further converted into plain text format by Lister. We then ran our TextMed system on each of these data sets by specifying the number of agents to implement during execution. Upon completion, TextMed outputs the data into an SQL database, and also transforms the data into visual graphs for analysis.

Figure 4 shows a heatmap describing each data set that we tested, with each experiment implementing 50 agents. The \( x \)-axis represents the top 50 keywords by global count, indicating which of the MeSH keywords appeared most fre-
Table 2: Description of our database tables

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary_Keyword</td>
<td>Contains all of the keywords used by Lister to generate the documents parsed by each agent.</td>
</tr>
<tr>
<td>Secondary_Keyword</td>
<td>Contains all of the MeSH keywords used by each agent to parse its subset of documents. Further contains global information for each keyword across all documents parsed, including total count, average calculated sentiment (prior to learning), final learned sentiment, and accumulated reward.</td>
</tr>
<tr>
<td>Document</td>
<td>Contains all documents that have at least one secondary keyword in their body of text. Further contains local information for each keyword that occurs in a particular document, including local count and average calculated sentiment (prior to learning).</td>
</tr>
<tr>
<td>Match</td>
<td>Contains the matching text for every secondary keyword match in each document, delimited based on the proximity value tested. Further contains the calculated sentiment for each match (prior to learning).</td>
</tr>
<tr>
<td>Learning</td>
<td>Contains the learning trials performed for every secondary keyword in each document in which it appears at least once. Order is maintained based on a parseOrder parameter (a ranking metric) in each keyword object, which increments by one after a learning trial is performed. Data is stored during each learning trial to show progression as the parseOrder increases. This data includes: local calculated sentiment (prior to learning), average global calculated sentiment (prior to learning), global learned sentiment (immediately after learning), and reward given per trial.</td>
</tr>
</tbody>
</table>

Figure 4: Heatmap showing the number of occurrences of the top 50 keywords with sentiment values for the three data sets used in our experiments with 50 agents. The x-axis represents the top 50 keywords, indicating which of the MeSH keywords appeared most frequently in the documents that were parsed. The y-axis represents global learned sentiment. The scale is a log normalized representation of a global count, with warmer colors indicating many keyword matches and cooler colors indicating few keyword matches. The dark blue color (the coolest color) under a specific sentiment value indicates the non-applicability of the reward, since a keyword can have only one sentimental value associated with it.

![Heatmap](image)

A: Muscular atrophy  
B: Alzheimer's disease  
C: Diabetes

Table 2: Description of our database tables

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary_Keyword</td>
<td>Contains all of the keywords used by Lister to generate the documents parsed by each agent.</td>
</tr>
<tr>
<td>Secondary_Keyword</td>
<td>Contains all of the MeSH keywords used by each agent to parse its subset of documents. Further contains global information for each keyword across all documents parsed, including total count, average calculated sentiment (prior to learning), final learned sentiment, and accumulated reward.</td>
</tr>
<tr>
<td>Document</td>
<td>Contains all documents that have at least one secondary keyword in their body of text. Further contains local information for each keyword that occurs in a particular document, including local count and average calculated sentiment (prior to learning).</td>
</tr>
<tr>
<td>Match</td>
<td>Contains the matching text for every secondary keyword match in each document, delimited based on the proximity value tested. Further contains the calculated sentiment for each match (prior to learning).</td>
</tr>
<tr>
<td>Learning</td>
<td>Contains the learning trials performed for every secondary keyword in each document in which it appears at least once. Order is maintained based on a parseOrder parameter (a ranking metric) in each keyword object, which increments by one after a learning trial is performed. Data is stored during each learning trial to show progression as the parseOrder increases. This data includes: local calculated sentiment (prior to learning), average global calculated sentiment (prior to learning), global learned sentiment (immediately after learning), and reward given per trial.</td>
</tr>
</tbody>
</table>

Figure 4: Heatmap showing the number of occurrences of the top 50 keywords with sentiment values for the three data sets used in our experiments with 50 agents. The x-axis represents the top 50 keywords, indicating which of the MeSH keywords appeared most frequently in the documents that were parsed. The y-axis represents global learned sentiment. The scale is a log normalized representation of a global count, with warmer colors indicating many keyword matches and cooler colors indicating few keyword matches. The dark blue color (the coolest color) under a specific sentiment value indicates the non-applicability of the reward, since a keyword can have only one sentimental value associated with it.
Alzheimer’s disease or muscular atrophy, it seems rational to expect lower sentiment scores in general.

4.2 Proximity Parameter

For our next set of experiments, we analyzed the proximity parameter to assess its effect on the results. Figure 5 shows the effects of proximity by comparing sentiment and reward values across three distinct proximity values. In these graphs, the x-axis represents the top 50 most frequently found keywords, the y-axis represents the sentiment values, while the scale represents the average reward given to a keyword at a particular proximity value. The reward was negated for better visual representation, so that warmer colors represent more optimal reward.

As explained previously, these generally negative sentiment scores are likely secondary to our biomedical data source combined with the way SentiStrength assigns positive and negative scores to certain words. Further, the fluctuation of scores follows from the number of words entered into sentiment analysis, as determined by proximity. Proximities of 5, 10, and 20 allow for 10, 20, and 40 words surrounding a keyword match to be analyzed by SentiStrength. This allows for considerable variability for the sentiment score, which could be influenced by both positive and negative words in small or large magnitudes. Now, considering the reward values (colors of the heatmaps), we observed that, in general, the reward values corroborate for individual keywords across the three tested proximity values. That is, if a specific keyword had a warmer color, indicating a higher reward, in one heatmap corresponding to a specific proximity value, it will, in general, have a warmer color in a heatmap corresponding to another proximity value. Upon a closer examination, we found that the sentiment values matched in all three tested proximity heatmaps in 25% of the keywords, they matched 2 out of 3 times in 63% of the keywords, and they did not match at all (with different sentiment value being assigned for a keyword for each tested proximity value) in 12% of the keywords. Thus, we concluded that the proximity is an important parameter to consider.

In order to further analyze the proximity parameter, we ran our TextMed program while testing proximity values between 1 and 25, which determines the amount of text around a keyword match to include in SentiStrength’s local sentiment calculation. All global values, such as global learned sentiment and accumulated reward, were stored individually between proximity values so that they did not influence each other. The heatmap in Figure 6 shows the relationship between the various proximity values and the average reward given to certain keywords. As with the previous heatmaps, the x-axis represents the top 50 keywords by global count. The y-axis represents the proximity values tested, while the scale represents the average reward given to a keyword at the proximity value tested. Again, the reward was negated, so that warmer colors correspond to more optimal reward. The highest concentration of warmer colors in Figure 6 occurs along the row where proximity equals 1, while the highest concentration of cooler colors appears along the row with the largest proximity value tested, 25. There further appears to be a gradual transition between proximities 1 and 25, with incremental values exhibiting lighter red hues that eventually turn blue. One can rationalize this result by first examining the way by which SentiStrength calculates local sentiment. SentiStrength uses different parts of speech in a string of text to grant positive and negative scores, which are then added together to form a composite score that TextMed records. When the proximity value was very small, there was a smaller chance for the string of text to contain any of the parts of speech that SentiStrength was looking for when calculating its score. Therefore, the calculated sentiment will likely remain around the same value for small proximity values, and conversely there is a larger possibility for variance as the proximity becomes larger. Further, using our reinforcement learning algorithm, smaller reward values were given when the difference between the current calculated local sentiment and historical local sentiment values were smaller. Thus, when the proximity value was small
and there was little fluctuation in the local calculated sentiment, the reward given during reinforcement learning was also small, resulting in the bright red rows in this heatmap.

### 4.3 Learning Parameters

<table>
<thead>
<tr>
<th>Number of agents in TextMed</th>
<th>Accumulated Utility per single agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>165</td>
</tr>
<tr>
<td>50</td>
<td>33</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
</tr>
</tbody>
</table>

For our next set of experiments, we analyzed the sentiment, reward and utility values obtained by TextMed. Figure 7 shows the individual agent utility per document for a system with (A) 10, (B) 50 and (C) 100 agents with proximity value of 10. We observed that with more agents, each agent being responsible for processing and analyzing fewer documents, there was less fluctuation in the agent’s utility value. We also noted that with a larger number of agents, each agent received less accumulated utility, as displayed in Table 4. In Figure 8 we can see that the reward for a keyword gets smaller (better) with each processed document, indicating that the learning process is able to improve over time. We also observe that the accumulated reward, calculated by adding rewards received after processing each individual document, is increasing over time as expected.

From Figure 9 we observed that the local sentiment (returned by SentiStrength) had high fluctuations, while the global learned sentiment obtained after the agents completed the reinforcement learning stage after processing each document, stabilized better with time. This indicates that the agents are able to learn the correct sentiment together while processing the documents. During our final set of experiments, we further analyzed the relationship between sentiment and the reward values. The heatmap in Figure 10 shows the relationship magnitude of the reward for different sentiment values assigned to various keywords. As before, the $x$-axis represents the top 50 keywords by global count. The $y$-axis represents the proximity values tested, while the scale represents the average reward given to a keyword at the proximity value tested, which was again negated, and thus warmer colors represent a more optimal reward. We observed that the keywords with more negative sentiment values tend to get better reward values. Since the reward indicates the learning progress, or how strongly the agents believe the keyword has been assigned a correct sentiment value, we assume that the agents were able to learn and stabilize their decision on the sentiment values for more negative words sooner. We also note that the final sentiment values obtained for 10, 50, and 100 agents were similar, and thus we do not produce all of the results from our experiments in this paper.

In summary, we conclude that our multi-agent system with learning scales well with the number of agents and that in general, it is able to efficiently and correctly learn the sentiment values of the keywords.

### 5. CONCLUSION

We developed a multi-agent system with reinforcement learning that is able to intelligently text mine and analyze the biomedical research articles. Software agents comprising our system collectively learn the sentiment pertaining to specific keywords as each agent processes its assigned subset of data. We conducted an experimental study to prove the concept of our system using the PubMed research articles for the keywords related to muscular atrophy, Alzheimer’s disease, and diabetes.

In the future, we plan to modify some of the details of Sen-
tiStrength to make it more appropriate for biomedical data, or as an alternative develop our own sentiment analysis tool aimed at biomedical data. We will also conduct more in-depth experiments, with each agent being tasked with processing and analyzing a data set from a different database, potentially stored on different physical machines. Finally, we would like to conduct experiments on other biomedical data, for example involving clinical data.

6. REFERENCES


A: Muscular atrophy  
B: Diabetes

Figure 10: Heatmap showing the relationship between sentiment values and the reward for the top 50 keywords with proximity 10 for different data sets in a system with 50 agents.