Peeps Finder: User-Validated Contact Attribute Extraction from the Internet

Daniele Moro
Boise State University
danielemoro@u.boisestate.edu

***Abstract*—Networking is an important part of a professional career, yet collecting and remembering personal information can be a cumbersome process. We introduce a new tool, *Peeps Finder*, that leverages various natural language processing and information retrieval techniques, such as noun phrase extraction, noun entity recognition, and relationship extraction, to find relevant contact attributes from internet searches. To avoid storing irrelevant personal information, we introduce a web-based user interface for attribute validation and system feedback. We evaluate our tool through a user study, demonstrating that *Peeps Finder* can be effectively leveraged to collect personal information from the internet, and that it has potential to be integrated into existing contact managers to aid professionals in building strong networks.**

I. Introduction

Professional networking is vital to career growth and success. A study of 279 professionals concluded that networking correlates with both objective and subjective measures of career success, as well as career growth, salary, and satisfaction [1]. Important factors include both strengthening existing connections, as well as constantly expanding one's network. Yet networking does not only entail ascertaining names, email addresses, phone numbers, and social media accounts. A successful professional network demonstrates "great range, reflecting a variety of strategically chosen professions, perspectives or personalities" [2]. Networking fundamentally involves constantly building a large graph of diverse individuals and their relationship to other individuals, organizations, and interests. A successful professional must attempt to continuously maintain and expand their network as they move forward in their career.

At the same time, studies in human psychology have found cognitive limits to the size of human social groups. The well-debated Dunbar Number places this limit at around 150 individuals [3]. Yet in an age where the internet connects countless individuals in endless ways, the modern professional must build a network far greater than a few hundred, and this network must grow quickly. This introduces a myriad of challenges, such as storing the vast information in this network and extracting relevant information to effectively leverage the network for career growth. As implied by the Dunbar Number, human memorization is inadequate, and so one must turn to technology to aid in both storage and retrieval of information.

Contacts managers such as Google Contacts can be effective in storing the information one already knows about a contact, but entering this information is time-consuming (each contact must be entered manually), and such tools are unable to retrieve information that a user omits or doesn't know. A powerful alternative is the Customer Relationship Management tool (CRM) which often offer "contact enrichment" features that automatically find personal information. However, these tools are often limited by their accepted input, database size, and/or types of contact attributes that they extract.

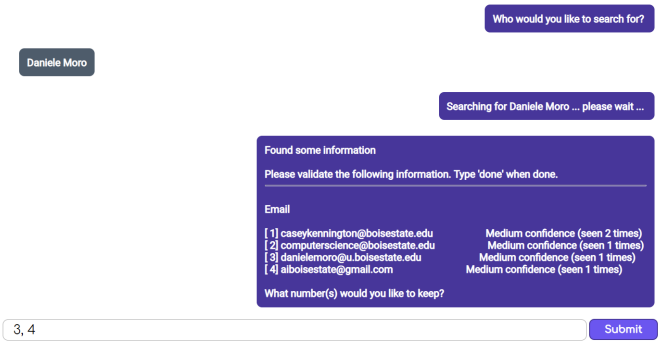

Figure 1. An example of the tool *Peeps Finder* being used to extract contact attributes. The user can enter the corresponding numbers to select the contact attribute value(s) they wish to keep. After responding to the displayed question, more contact attributes are are shown to the user for validation.

By leveraging recent advances in relationship extraction, and proven natural language processing techniques, relevant information about one's contacts can be extracted from the nearly limitless quantity of unstructured information found on the internet. We propose a novel tool called *Peeps Finder* that asks the user for a contact's first and last name, then crawls the internet for relevant web pages, and uses an ensemble of techniques such as noun phrase extraction, named entity recognition, and relationship extraction to automatically extract relevant personal information. This personal information is validated by the user through a web-based user interface (as shown in Figure 1) before being saved for later modification or retrieval.

In this paper, we contribute a novel tool for personal information extraction from the internet that has the potential to be integrated into existing contacts managers, as well as a web-based chat interface for validation, modification, and querying of collected personal information. We then explore the question "can *Peeps Finder* be effectively leveraged to collect relevant personal information from the internet?"

II. Background

Although small networks are important for career success and overall happiness [1], large networks are a significant

contributor towards a successful personal and professional life. “Large, heterogeneous networks have greater numbers of members who provide all kinds of support”, as well as “ companionship, minor services, major services, and emotional support” [4]. These large networks can increase the probability of one finding the right contact to advance in their profession, such as connecting to a potential investor, business partner, or employer. To effectively leverage this network, a professional must be able to find accurate emails and phone numbers to contact desirable individuals, ascertain the background and previous experiences of such individuals, and determine what organization and entities desirable individuals are connected to. Some of this information may be exchanged through conversation, but a successful professional must often resort to searching further information about a contact on the internet based off some limited starting information. This can take valuable time and effort that could be better allocated towards face-to-face networking. Furthermore, because of the cognitive limits of human social abilities [3], most personal information must also be recorded and then efficiently retrieved at the right moment.

This daunting task of collection and retrieval of personal information can be accomplished in part by leveraging contact managers such as Google Contacts, but as previously discussed, these simple tools are often inadequate for large and fast-growing networks because it quickly becomes cumbersome and slow to manually enter every piece of personal information. If one forgets to enter personal information, or if one does not know a certain piece of information, such as an email address, the contacts manager becomes much less useful. Enter the Customer Relationship Management tool (CRM), a software package targeted towards professionals in sales, marketing, or business, who must doggedly follow customer leads and build strong relationships based on detailed customer information [5]. Many CRMs offer features that allow a user to find the email address of an individual given the domain name of the individual’s organization [6]–[10]. Although an email-finding feature may be useful for initiating contact with potential customers, many users most likely do not know the domain name of a contact’s organization offhand, and even if the contact’s email is found, emails alone are not enough information to fill a network of connections.

A select few CRMs offer a “contact enrichment” feature which automatically find a variety of personal information given the name of an individual [7], [8], [10]. This can be extremely useful for networking, as one only needs to remember the name of a new contact, and the CRM will automatically find the relevant information to add to one’s professional network. However, these CRMs also exhibit several limitations. *Nobert* can only find information about a contact’s company, location, job role, and social media profiles, and it requires the email address of the contact to find this information [7]. *FullContact* can find information about a contact’s demographics, location, lifestyle, shopping habits, purchases, and finances, (more specific details are unavailable) but it is targeted toward use by businesses, not individual professionals [10].

Clearbit is perhaps the most comprehensive contact enrichment feature, collecting about 50 contact attributes types for an individual, but many of these attributes consist of social media handles rather than more potentially more useful contact attributes such as *occupation* and related individuals. *Clearbit* also requires the email of the individual to find such information, and it does not always return relevant information. Furthermore, since these technologies are proprietary, the inner workings of these tools are unknown. Speculation suggests that these CRMs are leveraging a large database of personal information largely acquired from partnerships with third party organizations. This means that in addition to being limited by their accepted input, and output contact fields, most of these CRMs are also limited by the size and freshness of their accrued data-set of records.

An alternative approach to finding relevant personal information lies in leveraging the copious amounts of unstructured personal data on the internet [11], [12] in tandem with information extraction and relationship extraction techniques. This offers the advantage of an almost limitless dataset that is constantly updated, but it also introduces the challenges of parsing unstructured text and validating the accuracy of the parsed information. Once clean text is obtained, relationship extraction models can extract relevant contact-entity relationships (contact attributes) in the text, such as workplace, occupation, education, etc. Existing relationship extraction tools include Stanford NER [13], MIT Information Extraction (MITIE) [14]–[17], OpenNRE [18]–[22], OpenIE [23], RESIDE [24], IEPY [25], and Context-Aware Representations for Knowledge Base Relation Extraction (CAKRE) [26].

Although both Stanford NER and MITIE acheived equivalent F1 scores of about 80% on the CoNLL 2002 NER task [14], [27], the remaining tools are difficult to compare as they all make use of different evaluation methods on a myriad of datasets. However, one important factor to compare is the size and quality of the types of relationships that the models extract. Stanford NRE’s range of extracted relationship types is relatively small, consisting of *Live_In, Located_In, Org-Based_In, and Work_For*. For MITIE, this number is slightly larger, consisting of 21 relationship types, but only 8 relate to a person (as opposed to an organization or event). OpenNRE can extract 52 different relationship types, with a significant portion regarding individuals. OpenIE does not require a pre-specified relation vocabulary, but sacrifices this with a subjectively worse ability to extract personal information (based on the author’s qualitative evaluation of OpenIE results on contact-related information). IEPY uses a rule-based strategy where the user must specify the relations to extract, making it difficult to define a sufficiently large set of relationship types for the purposes of this project. Finally, CAKRE offers one of the largest range of extracted relationship types, at 2,362 relations. All of these relationship extraction models, except Stanford NER and Open IE, leverage some form of word embedding, followed by a recurrent neural network such as an LSTM or GRU. Although these relationship extraction tools can effectively find entity relationships, they are not designed for contact attribute extraction from the noisy and heterogeneous data found through an internet search. However, for the purpose of this project, the capabilities of these tools

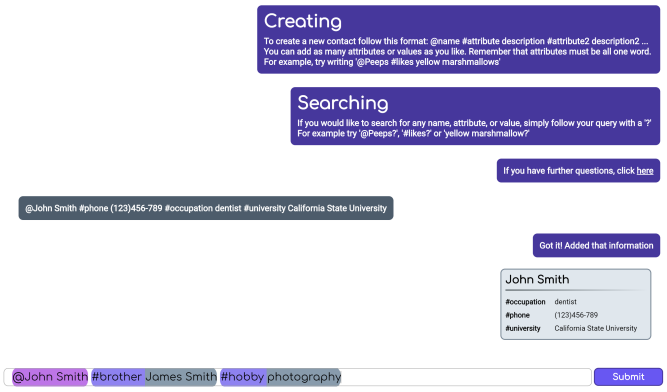

Figure 2. An example the *Peeps* web interface, and specifically how a user can use the tagging system to add or modify contact attributes.

were deemed satisfactory.

Chen et all. attempts to solve the problem of personal information extraction by using an unsupervised, rule-based information extraction system [28]. Their contact attribute extraction system relies on an approach of segmenting a web page into different styles of writing, as well as specific patterns in the structure of the page. Although their system achieved state-of-the art performance, they emphasize that "the total performance is still very low, which indicates the problem of Web personal information extraction is far from being solved and more work is needed." We expand upon this work by increasing the number of collected attribute classes from 16 to several thousand, merging multiple web pages to collect information about a single individual, and developing a user interface for validation of the collected personal information.

III. Methods

Our goal was to build an easy-to-use web-based tool for automatic collection of an individual's personal information. The only input required for *Peeps Finder* is the first and last name of the individual, and the output consists of a list of contact attributes and corresponding values. To accomplish this, we built a pipeline that leverages multiple natural language processing and information retrieval tools to provide the best results for the end user. *Peeps Finder* was programmed using Python 3.6, and it leverages a separate web interface called *Peeps* for user interaction. *Peeps Finder* searches for the contact using Google, crawls the relevant web pages, parses and cleans text from the resulting HTML pages, and then sends these cleaned documents through six components, each of which extracts certain contact attributes. These six components are the *Email Finder*, *Phone Finder*, *TFIDF*, *Named Entity Recognizer*, *Noun Phrase Extractor*, and *Relationship Extractor*. Figure 3 illustrates how all of these components, along with the web interface, interact in our solution.

Although *Peeps Finder* can be used through a command-line interface, we also built a web-based chat user interface called *Peeps*. This simple interface has the system's prompts appear on the right of the screen, and the user can type their response on the bottom of the screen. After submitting their response, a user will see their response appear on the left of the screen, allowing the user to see a history of the chat. Figure 2 shows the look of the interface. *Peeps* was coded using PHP, HTML, CSS, JavaScript, and jQuery. To enable a faster and more dynamic interface, we leverage the Asynchronous JavaScript and XML (AJAX) technique [29], dramatically improving the responsiveness of the interface and facilitating a more natural interaction. Once personal information is entered into the system, the information is stored on a MySQL database, and the user can view all of their contacts on a separate screen labeled *Contacts*.

Peeps also features a fast and simple tagging system for manually entering and modifying personal information. This tagging system was inspired by Slack [30], Todoist [31], and Augi [32]. The user can type an @ symbol followed by a name, followed by an # symbol, followed by the attribute of the contact, followed by the value of the attribute to add information to a contact. For example, "*@John Smith #phone 123-456-7890*" would add the phone number 123-456-7890 to the contact named John Smith. If John Smith does not exist in the database, the contact is created. If the attribute *#phone* already exists for John Smith, the old value is replaced with the new to allow for modification. The user can also specify multiple attributes and values in one phrase, such as "*@John Smith #phone 123-456-7890 #occupation engineer*", to quickly add or modify information about one contact. Finally, the user can query information from *Peeps* by using a *?* symbol at the end of a contact name, attribute, or value. For example, to query information about John Smith, the user would enter "*@John Smith?*", and to query all contacts that contain *engineer* as a value, the user would enter "*engineer?*".

Peeps Finder augments *Peeps* by allowing the user to enter the keyword *find*, prompting an interaction with *Peeps Finder* where information about any person of the user's choosing (including people that are not in the contact list) is extracted and added to the user's contact list. A future version of *Peeps Finder* would automatically extract information about all contacts in a user's contact list, and then query the user for validation at a later time. However, to facilitate the evaluation of our tool, we require the user to enter the keyword *find* to initiate a search. After prompting *Peeps Finder* within the *Peeps* interface, the user is asked to provide the first and last name of the individual to be searched. After simple input validation, the *Peeps* web server communicates with a separate *Peeps Finder* process, which initiates the search and controls all user interaction until the completion of the search and validation.

After receiving the first and last name of the contact to search, *Peeps Finder* proceeds by performing an internet search of the full name. This search is done through Google, and leverages the *google-search* Python package [33]. We store the top 20 web results, and download their respective HTML pages using the urllib standard Python library. We then use BeautifulSoup [34] to parse these HTML pages for other pages in the same domain to crawl. This is an important step, as much personal information can be found on sites with multiple pages, such as a personal website containing an *About* page, *Projects* page, and more. The HTML of the top 10 same-domain pages for each of the top results is downloaded. After

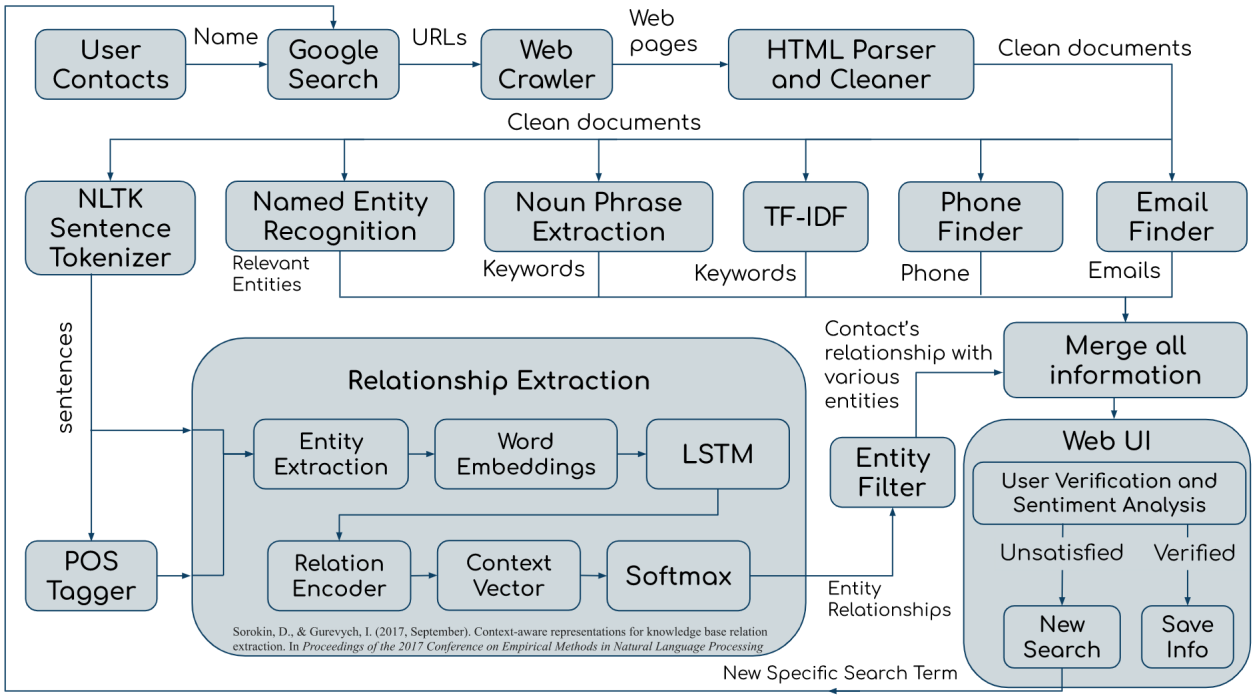

Figure 3. The pipeline for *Peeps Finder*. The name of a contact is searched, and the relevant HTML pages are parsed to extract clean documents. These documents are leveraged by several components, such as the *Email Finder*, *Phone Finder*, *TFIDF*, *Named Entity Recognizer*, *Noun Phrase Extractor*, and *Relationship Extractor* to extract relevant contact attributes and their corrresponding values. This data is presented to the user for validation, with an option to repeat the process using a more specific search term.

downloading all of the above pages, BeautifulSoup is used to extract all of the text found within the *p* tags of the HTML, as we found that the text within the paragraph tags was often the cleanest and most useful text. Although this process removes much information from the web search, we found that the most relevant information is still extracted. All of the cleaned text, separated into documents from the various web pages, is saved to a file for later use. Saving the cleaned text allows for faster retrieval if a later user of *Peeps Finder* searches for the same person.

The *Email Finder* and *Phone Finder* are fairly straightforward. As hinted by their names, the components either search for relevant phone numbers or emails. This is done by first merging the cleaned text from all of the documents into one string. Regular expressions are then used to extract and count all instances of emails and phone numbers that can be found. We use the following regular expressions to accomplish this:

- Email address regular expression [35]

```
[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+
@[a-zA-Z0-9](?:[a-zA-Z0-9-] {0,61}
[a-zA-Z0-9])?(?:\.[a-zA-Z0-9]
(?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*
```

- Phone number regular expression [36]

```
\D?(\d{3})\D?\D?(\d{3})\D?(\d{4})
```

We found that these particular regular expressions worked for most of the emails and phone numbers in the cleaned documents. We decided to use regular expressions because they are straightforward to implement and can detect a large amount of deviations in how a phone number or email may be recorded. However, it is important to note that these regular expressions will not detect all variations of phone numbers and emails, and no validation is done on email address.

The *TFIDF* and *Noun Phrase Extractor* components find keywords related to the individual being searched. TFIDF is a well-used technique in information extraction that gives every word in a group of documents a score of importance. TFIDF stands for term frequency-inverse document frequency, and, as implied by the name, it places more importance to words that appear often within a certain document, but less importance to words which appear often in every document [37]. This allows common stop words to receive a low importance, while driving up the importance of words unique to a document (such as important personal information). There are a myriad of variations and implementations of TFIDF, but we chose to use the Scikit-Learn implementation, specifically their CountVectorizer and TfidfTransformer, as they are straightforward to use in Python [38]. After sorting every word in each document according to the TFIDF importance, the most common 500 words in the English language (as determined by Google's Trillion Word Corpus) are filtered out to remove meaningless keywords [39]. We also remove words from a manually created blacklist containing words such as "javascript" and "tweet" that were included after several iterations of testing. The top 30 of the resulting keywords are collected. The *Noun Phrase Extractor* component also collects relevant contact keywords, but it uses TextBlob's Noun Phrase extraction tool [40] to accomplish this. TextBlob leverages a simple rule-based algorithm with part of speech tags to extract these noun phrases. Noun phrases

are useful because they often contain the subjects covered in the text. By counting all instances of noun phrases and sorting them based on their frequency, the most common noun phrases can convey relevant personal information. Similarly to the *TFIDF* component, the most common English words and blacklisted words are removed from the top 20 noun phrases. At the end of this process, we have a list of 50 keywords that are related to the searched contact.

Although the keywords related to a contact can be useful, these keywords lack their corresponding contact attribute (we know the value of the attribute, but not the name of the attribute itself). Although the contact attribute can be manually labeled by the user, a more useful tool would determine both the contact attribute and its corresponding value. To accomplish this, we leverage both the *Named Entity Recognizer* component and the *Relationship Extractor* component. The *Named Entity Recognizer* component uses the spaCy named entity recognition feature [41], which uses various statistical models on part of speech tags and dependency parses to extract the relevant entities. spaCy can recognize 18 different types of entities, providing both the contact attribute (the entity class) and the value of the attribute (the instance of the entity). These 18 entity types include *person*, *organization*, *nationality*, and *location*. We extract all named entities found in all documents, and we sort them based on frequency, using only the top 30 named entities. Many of these entity types were renamed to be more clear for the user, such as "person" being renamed to "knows" (as in knows person x), and the following named entity classes were disregarded: *number*, *important date*, *important time*, and *family name* because they were either not useful or often incorrect.

One of the most powerful ways for finding relevant personal information is by making use of relationship extraction tools. After qualitatively analyzing the relationship extraction models discussed in the background section, we decided to use CAKRE [26] for *Peeps Finder* as it offers one of the largest ontologies, is compatible with Python 3 (which is required for other components of the project), and is relatively straightforward to replicate the results on a Windows 10 machine. CAKRE boasts precision and recall scores of over 80% on the top 6 relations, and after qualitative tests of relationship extraction on a variety of documents containing information about several individuals, we found CAKRE to return relevant and acceptable entity relationships. CAKRE requires clean, part-of-speech tagged sentences as inputs, which we collect from the NLTK Sentence Tokenizer [42] and spaCy part-of-speech tokenizer [41]. It extracts the named entities and noun chunks using Stanford CoreNLP to find all entity pairs in the sentence. It embeds every entity pair, and passes this embedding through a Long Short-Term Memory (LSTM) [43] model, which generates a relation vector. These relation vectors are combined into a context vector, which is finally used to generate a graph of how every entity relates to every other entity. The relevant contact-entity relations can then be extracted from this graph.

The emails found by the *Email Finder*, phone numbers found by the *Phone Finder*, keywords collected by the *TFIDF* and *Named Entity Recognizer* components, and the contact-entity relationship pairs extracted by the *Named Entity Recognizer* and *Relationship Extractor* components are all collected into one dictionary consisting of contact attributes and a corresponding list of attribute values. By combining the email, phone, keyword, 14 named entity classes, and 2,362 relationship types from CAKRE, *Peeps Finder* is capable of extracting 2379 contact attributes types. Each attribute found for a searched contact is given a loose confidence rating based on the number of times that contact attribute has been seen, and then presented to the user.

If *Peeps Finder* were to automatically place these collected attributes and values into a user's contact list, we suspect that the user might be displeased, as incorrect or incoherent information may be recorded. From our evaluation of *Peeps Finder*, we conclude that the tool is still not providing perfect information about a contact (see section IV), so we augment our existing tools with user validation and feedback to provide satisfactory results. Each of the collected attributes, along with its potential values, is shown to the user one attribute at a time using the *Peeps* web interface. The user is then asked to validate the information by entering one or multiple numbers corresponding to the values they wish to keep (Figure 1).

After marking which values to keep for every attribute, the user is presented the summary of validated personal information and asked for a written evaluation of the data. The user's response is analyzed using TextBlob's sentiment analysis tool [40], which returns a number representing the polarity of the user's sentiment. If user provides a positive review, the information is saved to the user's contact list for later retrieval and/or modification. If the user instead provides a negative review, they are asked if they would like to repeat the search with a more specific search term. The user is also supplied with a recommendation of more specific search terms based off of the information they previously validated about the contact. If the user agrees to committing a new search and selects or provides a new search term, the process repeats with the new search term until the user is satisfied. The user is provided the option to execute a more specific search in order to decrease the probability that irrelevant results (such as a person with the same name) appear in the search results. Once satisfied with the results, the user is free to use the *Peeps* web interface to modify, add, or query personal information. Although we aim to capture the majority of personal information a user would record, manually adding or modifying information can be important if the user would like to store information not returned by *Peeps Finder* or private information only known by the user.

IV. Evaluation

In order to answer the question "*Can Peeps Finder be effectively leveraged to collect personal information from the internet?*", we conducted a user study with 5 human participants who judged the effectiveness of the tool. This was a necessary component of evaluation because there are no obvious automated metrics to evaluate the quality of the collected contact attributes and the associated user interface. The participants were asked to use *Peeps Finder* to collect

information about two to three friends, acquaintances, or colleagues. Each participant spent about 20 minutes interacting with *Peeps Finder*, and the participants were not financially compensated.

After using *Peeps Finder*, each participant answered the following questions:

1) Who are the people you sought information about? Short answer response.
2) Were you satisfied by the initial results returned by the bot? Scale from 1 (incoherent) to 10 (accurate and detailed).
3) Were you satisfied by the results returned by the bot after an iteration of feedback? Scale from 1 (incoherent) to 10 (accurate and detailed).
4) Would you use this tool to find information about your contacts? Scale from 1 (would never use) to 10 (would use every day)
5) Would you consider integrating this bot into your existing contacts manager? Short answer response.
6) Please provide any suggestions for further improvement. Short answer response.

According to responses to the first question, participants sought information about friends, acquaintances, or colleagues, who had varying levels of an online presence. The average score from the initial extracted personal information was 6.6, showing that *Peeps Finder* returns decently accurate and detailed information, and it is not incoherent (Figure 4). The average score after an iteration of feedback (conducting a more specific search term) was 8.2, highlighting significant improvement in resulting personal information accuracy and detail (Figure 5). Many participants augmented their narrower search term by adding a key word after the name of the contact, such as a city or workplace. Furthermore, the polarity of the feedback extracted by TextBlob [40] after users validate information in relation to both the first and second searches of a contact increased on average by 1.12, where the polarity of the sentiment ranged of -1 (negative) to 1 (positive). This may indicate that person disambiguation is a significant challenge, and that specifying a narrower search term removes irrelevant results, drastically improving user satisfaction.

Even with imperfect personal information returned by *Peeps Finder*, the average score of question four was 7.0, with most responses clustering towards using the tool very often (Figure 6). This highlights the effectiveness of *Peeps Finder*. When asked in short answer form if they would consider integrating *Peeps Finder* into their existing contacts manager, all of the participants indicated that they would, with one participant agreeing that it would save much time, and another participant commenting on the usefulness of the tool.

When asked to provide further suggestions, four of the participants suggested improvements to the user interface, such as showing some visual indication when the user is expected to wait for a response, more efficient methods of selecting the attribute values that the user wishes to keep, or including a picture of the person found. Two of the participants mentioned that the tool should be faster, and three of the participants mentioned that results about multiple people appeared in their

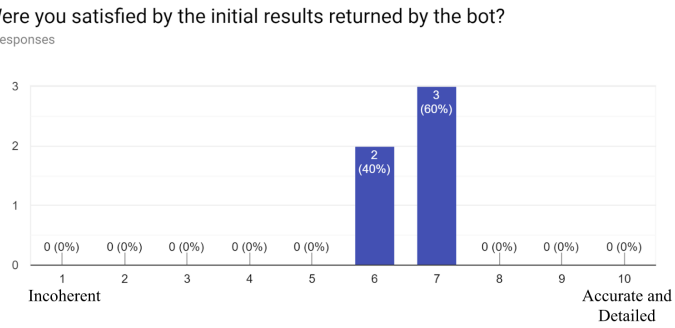

Figure 4. Results from a user study of 5 participants, where the participants answered the question "Were you satisfied by the initial results returned by the bot?". Responses were recorded on scale from 1 (incoherent) to 10 (accurate and detailed)

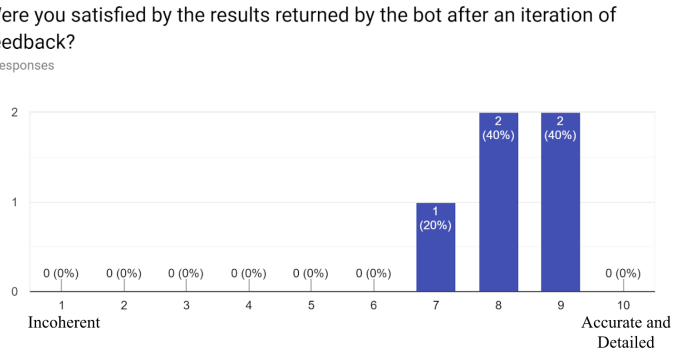

Figure 5. Results from a user study of 5 participants, where the participants answered the question "Were you satisfied by the results returned by the bot after an iteration of feedback?". Responses were recorded on scale from 1 (incoherent) to 10 (accurate and detailed)

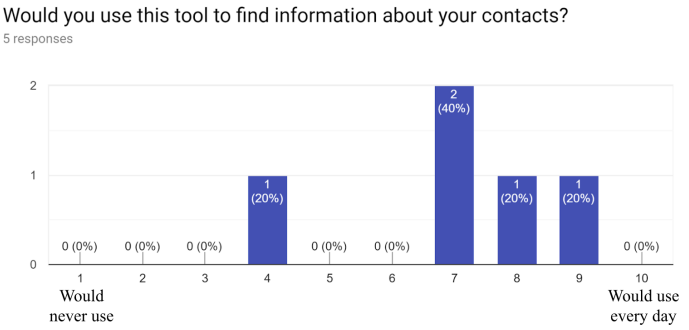

Figure 6. Results from a user study of 5 participants, where the participants answered the question "Would you use this tool to find information about your contacts?". Responses were recorded on scale from 1 (would never use) to 10 (would use every day)

results. One of the participants suggested allowing the user to add a key word in addition to the contact's name as so to narrow the initial search and avoid these irrelevant results.

Our results substantiate the claim that *Peeps Finder* can be effectively leveraged to collect personal information from the internet. Although *Peeps Finder* has potential for improvement, results from the the user study, especially responses to questions three and four, demonstrate that the collected personal information is both useful and relevant. Furthermore, this evaluation paves the way for further improvement of *Peeps Finder* in the areas of large-scale person disambiguation, user

interface design, and relationship extraction.

V. Conclusion

To aid professionals in networking, efficient and intelligent tools to manage and expand personal information must be developed. *Peeps Finder* offers automated contact attribute extraction by leveraging unstructured data collected from the internet, in tandem with a myriad of information extraction and natural language processing techniques, such as noun phrase extraction, named entity recognition, and relationship extraction, to extract 2379 different contact attributes with multiple options for their values. To augment the extracted personal information, *Peeps Finder* leverages a web-based user interface called *Peeps* for user validation, as well as personal information modification, addition, and querying. A user study of the tool shows that *Peeps Finder* can be effectively leveraged to collect personal information from the internet, and that the tool has great potential to be integrated into contacts managers to provide a semi-automated expansion in the quantity and quality of stored personal information.

Future work lies in user interface design, the continued addition of relationship extraction models, and large-scale person disambiguation. A more effective and faster user interface may conduct searches and extract personal information for all of the contacts in a user's network. After this search is complete, the tool may send a notification to the user, asking them to validate the information. Furthermore, since the user does not always know if some piece of personal information is accurate, the user interface may make use of multiple labels, such as "correct", "not sure", and "sounds about right". Additional improvements to the design of the user interface could facilitate faster and more effective correction during contact validation [44]. Applying multiple relationship extraction models could also improve the accuracy and detail of the collected information, combining the strengths of various relationship extraction models. Finally, one of the most significant opportunities for improvement lies in person disambiguation, and more broadly in large scale entity reference resolution [12] [45]. Continued development of relationship extraction models, combined with unsupervised clustering approaches may be viable solutions.

References

[1] H.-G. Wolff and K. Moser, "Effects of networking on career success: a longitudinal study." *Journal of Applied Psychology*, vol. 94, no. 1, p. 196, 2009.

[2] S. Bodell and A. Hook, "Using facebook for professional networking: a modern-day essential," *British Journal of Occupational Therapy*, vol. 74, no. 12, pp. 588–590, 2011.

[3] R. Dunbar and R. I. M. Dunbar, *Grooming, gossip, and the evolution of language*. Harvard University Press, 1998.

[4] B. Wellman and M. Gulia, "Where does social support come from? the social network basis of interpersonal resources for coping with stress."

[5] F. Buttle, *Customer relationship management*. Routledge, 2008.

[6] Hunter, "Find email addresses in seconds hunter (email hunter)." [Online]. Available: https://hunter.io/

[7] V. LLC, "Email finder - find anyone's email address." [Online]. Available: https://www.voilanorbert.com/

[8] Clearbit, "Automatically enrich your records and workflows." [Online]. Available: https://clearbit.com/enrichment

[9] G. Inc, "An outbound sales platform to get new customers faster." [Online]. Available: https://www.growbots.com/

[10] FullContact, "Our data," Feb 2019. [Online]. Available: https://www.fullcontact.com/our-data/

[11] D. DiLillo and E. B. Gale, "To google or not to google: Graduate students' use of the internet to access personal information about clients." *Training and Education in Professional Psychology*, vol. 5, no. 3, p. 160, 2011.

[12] G. S. Mann and D. Yarowsky, "Unsupervised personal name disambiguation," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics, 2003, pp. 33–40.

[13] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[14] M. N. Group, "Mitie: Mit information extraction," Feb 2019. [Online]. Available: https://github.com/mit-nlp/MITIE

[15] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1755–1758, 2009.

[16] P. S. Dhillon, D. P. Foster, and L. H. Ungar, "Eigenwords: Spectral word embeddings," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 3035–3078, 2015.

[17] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural svms," *Machine Learning*, vol. 77, no. 1, pp. 27–59, 2009.

[18] T. Gao, X. Han, S. Cao, L. Tang, Y. Lin, and Z. Liu, "Opennre," 2019. [Online]. Available: https://github.com/thunlp/OpenNRE/

[19] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 2124–2133.

[20] Y. Wu, D. Bamman, and S. Russell, "Adversarial training for relation extraction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1778–1783.

[21] T. Liu, K. Wang, B. Chang, and Z. Sui, "A soft-label method for noise-tolerant distantly supervised relation extraction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1790–1795.

[22] J. Feng, M. Huang, L. Zhao, Y. Yang, and X. Zhu, "Reinforcement learning for relation classification from noisy data," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] U. of Washington and D. Indian Institute of Technology, "Open ie," 2018. [Online]. Available: https://github.com/dair-iitd/OpenIE-standalone

[24] S. Vashishth, R. Joshi, S. S. Prayaga, C. Bhattacharyya, and P. Talukdar, "Reside: Improving distantly-supervised neural relation extraction using side information," *arXiv preprint arXiv:1812.04361*, 2018.

[25] R. Carrascosa, J. Mansilla, G. Berrotaran, F. Luque, and D. Moisset, "iepy," Oct 2016. [Online]. Available: https://pypi.org/project/iepy/

[26] D. Sorokin and I. Gurevych, "Context-Aware Representations for Knowledge Base Relation Extraction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2017, pp. 1784–1789.

[27] S. N. Group, "Stanford nlp named entity recognition results," Jul 2009. [Online]. Available: https://nlp.stanford.edu/projects/project-ner.shtml

[28] Y. Chen, S. Y. M. Lee, and C.-R. Huang, "Polyuhk: A robust information extraction system for web personal names," in *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, 2009.

[29] J. J. Garrett *et al.*, "Ajax: A new approach to web applications," 2005.

[30] S. Technologies, "Slack," Aug 2013. [Online]. Available: https://slack.com/

[31] Todoist, "Todoist," Jan 2007. [Online]. Available: https://todoist.com

[32] J. Ronnow, C. Kennington, and D. Kondratyuk, "Augi." [Online]. Available: https://helloaugi.com/about

[33] A. Hseb, "google-search," May 2017. [Online]. Available: https://pypi.org/project/google-search/#description

[34] L. Richardson, "Beautiful soup," Jan 2019. [Online]. Available: https://pypi.org/project/beautifulsoup4/#description

[35] D. Tools, "Regex to match a valid email address." [Online]. Available: https://www.regextester.com/19

[36] L. O'Donnell, "Phone number regular expression." [Online]. Available: http://regexlib.com/REDetails.aspx?regexp_id=45

[37] "A single-page tutorial - information retrieval and text mining." [Online]. Available: http://www.tfidf.com/

[38] K. Ganesan, "Keyword extraction with tf-idf and python's scikit-learn," Aug 2018. [Online]. Available: http://kavita-ganesan.com/extracting-keywords-from-text-tfidf/

[39] J. Kaufman, "google-10000-english," Mar 2018. [Online]. Available: https://github.com/first20hours/google-10000-english
[40] TextBlob, "Textblob," Mar 2019. [Online]. Available: https://textblob.readthedocs.io/en/dev/
[41] E. AI, "spacy," Mar 2019. [Online]. Available: https://spacy.io/
[42] S. Bird, "Natural language toolkit," Apr 2019. [Online]. Available: https://pypi.org/project/nltk/
[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[44] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum, "Interactive information extraction with constrained conditional random fields," in *AAAI*, vol. 4, 2004, pp. 412–418.
[45] J. Xu, Q. Lu, M. Li, and W. Li, "Web person disambiguation using hierarchical co-reference model," in *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, 2015, pp. 279–291.