From Brooklyn Barbers To Movie Stars: Using Introductions To Construct Embeddings Of People

Philippe Laban
UC Berkeley
phillab@berkeley.edu

John Canny
UC Berkeley
canny@berkeley.edu

Marti A. Hearst
UC Berkeley
hearst@berkeley.edu

Abstract

People are central to narrative texts such as news; a given news story can contain mentions of hundreds of distinct individuals, some famous and some not. We introduce a novel method for representing person entities in text, by using *introductions* across a large news corpus as input to an embedding model. We demonstrate the utility of the model, Per2vec, for the task of named-entity linking of people mentions to a knowledge base, showing that the method is particularly suited for the under-explored task of recognizing when a person *does not* appear in the knowledge base. We compare Per2vec's results to a state-of-the-art system on a challenging dataset and outperform it, and present two case studies.

1 Introduction

When reading text, proper names are used to identify and differentiate people. Unfortunately, names are not uniquely assigned, and ambiguity arises. When looking at a large corpus of text, such as millions of news articles,[1] ambiguities turn into collisions: distinct individuals share the same name. The field of Named Entity Disambiguation aims to resolve such ambiguities, to avoid confusing one person for another.

Named Entity Disambiguation is commonly performed by *linking*: an external knowledge base contains a large set of people, and given a named-entity occurrence, it is assigned (linked) to an entry in the knowledge base. However, no knowledge base is complete, and when an entry does not exist for an named-entity, a *nil* link is assigned. Assigning *nil* links can be difficult if a person has a common name, because there can be many homonyms present in the knowledge base, and all have to be ruled out in order to produce a *nil* link.

[1]The citation describing the news dataset and story generation technique has been omitted for anonymous review.

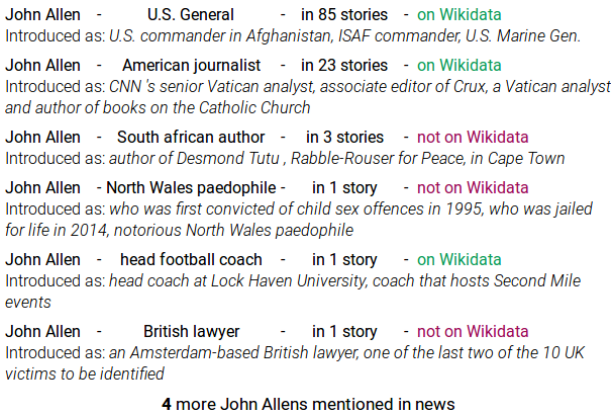

Figure 1: Identities of distinct John Allens, Wikidata presence, frequency in stories in a large news corpus.

For example, in a news article about the MH17 plane crash in 2014, John Allen is mentioned — he was a British lawyer who was a victim of the crash. But other homonymous John Allens appear in other news stories, some at higher frequencies, as shown in Figure 1. In this case, John Allen the lawyer is not in our knowledge base (Wikidata), and it is important to produce a *nil* link, to avoid assuming the death of the wrong person.

We propose to build a *person embedding, Per2vec*: a continuous vector that represents the person in its context. This is analogous to word vectors, which contain semantic information about discrete word objects. In state-of-the-art NLP models, the deep neural network sees the words only through their continuous word vectors, and exploits the semantics in the vector to augment its knowledge. Similarly, a person embedding represents the person in text, and can be used by subsequent tasks, such as named-entity disambiguation.

The "quality" of the embedding, and what semantic information it embeds is essential. Changing only the word embedding for a given model can increase accuracy on named-entity recognition

by up to 10% (Pennington et al., 2014). Therefore a central question addressed by this work is understanding what semantic information should go into the embedding of a person. The most popular word embeddings methodologies are Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014), which rely on co-occurrence of words in small neighborhoods, for large corpora of text. These methods work because similar words often occur in the same contexts (neighborhoods). More modern word embedding methods rely on neural networks working on an annex task, such as machine translation (McCann et al., 2017).

For people in news stories, sparsity is an obstacle, as most people are only mentioned a handful of times, thus not providing a broad enough context for standard embedding. We introduce a method that makes use of a priori knowledge of semantic information relevant to people: attributes such as occupation, gender, and citizenship. Our method exploits this knowledge to build embeddings specific to people.

We also introduce the use of *introductions*, very precise information used by news articles and other narrative forms to describe who a person is and what their role is. We expect a person's introduction to contain information needed to determine who they are and what their role is, as well as whether the person should be present on in a knowledge base. If a person is introduced as "a local barber in Brooklyn," the person is unlikely to have a Wikidata entry, but if they are introduced as "a famous movie star," they should have one.

Our contributions include:

1. A novel method, Per2vec, for embedding people from text, based on how they are introduced,

2. A named-entity linking system that is highly accurate at deciding that a person is *not* in a knowledge base,

3. An approach to named-entity linking using Per2vec that outperforms a state-of-the-art system on an open-sourced challenge dataset,

4. A downloadable model and simple code for researchers to embed and link people.[2]

[2]Link has been omitted for anonymity.

2 Related Work

We propose to embed people into continuous vectors, following the success of word vector embeddings. Many extensions to word vectors have been suggested: dealing with multi-sensed words, making the embeddings work across languages, and embedding phrases, entities and concepts. Several groups have already proposed building entity embeddings leveraging Wikipedia text. Finally, our embeddings are useful at the task of named entity disambiguation.

2.1 Word Vector Extensions

Word vectors are trained from data, usually in a single language, and are intended to capture semantic information. Recent work (Faruqui and Dyer, 2014; Upadhyay et al., 2016; Wick et al., 2016) has focused on building multilingual embeddings to have several languages in a common semantic space. But because words can have multiple senses, other groups have proposed methods to build word embeddings where words can have as many vectors as they have senses (Trask et al., 2015). These vectors can then be used for word sense disambiguation (Navigli, 2009). Finally, some work has looked at embedding phrases, entities and concepts more complex than single words within the same embedding space as words. Cho et al. (2014) learn phrase embedding representations as part of a Machine Translation pipeline, and Sherkat and Milios (2017) propose to embed Wikipedia anchors as special tokens in the vocabulary to represent entities and concepts.

In contrast, as the semantic information we aim to embed is not similar to word semantic information (word co-occurrence), we do not bridge our person embeddings with word embeddings.

2.2 Embeddings of People and Entities

Building embeddings for named entities has been explored in diverse ways. Many systems rely on Wikipedia and structured knowledge graphs such as Wikidata (Suzuki et al., 2016; Sherkat and Milios, 2017). These systems exploit the wealth of information available in a large external knowledge source, so that it can be used in context with new text. When finding an entity in a new document, its embedding can be used, being enriched by the information available in the knowledge graph. This comes with some complications: first of all, no knowledge graph is complete (most people men-

tioned in news are not on Wikipedia). Secondly, in order to use an externally generated embedding, perfect disambiguation is assumed: when John Allen is mentioned in a document, in order to use the correct entity vector, one must know which entry it corresponds to in the knowledge graph.

2.3 Named Entity Disambiguation

According to Hachey et al. (2013), named entity disambiguation is the grouping of 3 subfields: cross-document co-reference resolution (CDCR), where the goal is to link mentions of the same entity across documents (Bagga and Baldwin, 1998), Wikification, where given a piece of text, the task is to connect groups of words to Wikipedia pages (concepts as well as entities) (Mihalcea and Csomai, 2007), and finally named-entity linking (Bunescu and Paşca, 2006), where each named-entity in a text document is linked to an entity in a knowledge graph. If the entity is not in the knowledge graph, the link predicted must be *nil*.

Hoffart et al. (2011) built one of the first datasets for named-entity linking, based on the CoNLL 2003 data, often used for in the task of Named Entity recognition (Sang and Meulder, 2003). The data is made of 1400 news articles from Reuters, with a total of 35,000 entities manually linked, 7,000 of which (20%) are *nil* links. This dataset has since become a standard for benchmarking entity linking systems (Cornolti et al., 2013).

3 Introducing Introductions

A common embedding strategy is to define the "context" as a window of words (to the left and to the right) at each mention of the entity. When building the representation of a *person*, special kinds of textual context can be used. These include: quotes by the person, references to the person, representations of other co-occurring entities, and structured knowledge about the person.

Contingent on what text is used to embed the person, different embeddings are achieved. For example, using only knowledge base information implies the embedding obtained for a given person does not vary across contexts, although the role of the person might differ from story to story. Relying on quotes means people that were silent and did not produce direct quotes cannot be embedded.

The scripts for plays often contain stage directions that introduce a character's name, their role (her servant), their current state (entered in a hurry), or an action to take (picks up the phone). Equivalently, in many domains such as news articles, reports, and formal documents, when a person is introduced, several linguistic mechanisms add relevant personal information. We propose that such *introductions* are excellent sources for representing people:

- They are more focused than using a generic "window context": they only include the subpart of the text that refers to the person, and
- They can include acontextual information about the person, but also explain *why they are in the story*, which helps deduce their role in the context.

3.1 Definition of Introduction

Extraction of introductions is based on several linguistic patterns over the dependency parse of sentences where an person is mentioned. The set of dependency parse tags we use are from 'nlp4j' (Choi and McCallum, 2013), a standard for dependency parsing. We use the spaCy NLP toolkit for dependency parsing (Honnibal and Montani, 2017). The extraction patterns for introductions for PERSON are the following:

1. **P1 - appositions and relative clauses**: Child subtrees to the PERSON mention that are: appositional modifiers (*appos*), or relative clause modifiers (*relcl*).
2. **P2 - Compounds**: Words that are tagged as a compound of the PERSON (*compound*)
3. **P3 - adjectives and prep**: Child subtrees to the PERSON mention that are: adjectival modifiers (*amod*), or prepositional modifiers (*prep*)
4. **P4 - inverted apposition**: if PERSON is an appositional child, we extract the tree of its parent, and remove the apposition

Table 1 shows examples of each pattern.

Because we do not use a co-reference system, only introductions on direct person mention are extracted, while introductions on pronouns and other indirect mentions are missed. Because the extraction patterns are broad, errors or irrelevant extractions occur. We found an error rate of 8% when assessing all introductions extracted from a

Type	Example	Subtype
P1	**Euclid Tsakalotos**, *who is also a top negotiator in talks with Athens' lenders*, told	relcl
P1	But **Jeroen Dijsselbloem**, *head of the Eurogroup of euro zone finance ministers*, said	appos
P2	His boisterous mood belied his tough talks with *German Chancellor* **Angela Merkel** and *French President* **Francois Hollande** on Wednesday, and *European Commission President* **Jean-Claude Juncker** on Thursday.	N/A
P3	*The normally jovial* **Juncker** refused to take a telephone call from Tsipras after that and all but accused him of duplicity.	amod
P3	Tsipras sounded bitter and wounded after the creditors, led by **Christine Lagarde** *of the International Monetary Fund*, ...	prep
P4	*The Greek interior minister*, **Nikos Voutsis** insisted...	N/A

Table 1: Example extracted introductions for each pattern. Introduction in italics, person in bold.

Category	#	Example Introductions
Work	254	a paleontologist at the Virginia Museum of Natural History
Title	131	Mr., Ms., Sir, Dr.
Event	59	who plans to add two tours and open a chocolate school in Washington
Geo origin	34	New Zealand fighter, St. Louis-based broker
Description with Commentary	20	who is a popular and very humorous novelist, alleged financier of the genocide
Geo current	17	in Hangzhou, in Moscow
Familial	10	The bride's brother, Richard's grandson
Other / noise	12	Super Samoan (nickname)

Table 2: Types of introductions, with examples, from a random sample of 100 news articles.

news story made of 87 news articles (31 out of 404 introductions). It is common for a document to produce several introductions for a single person, in which case all introductions are kept.

3.2 Statistics of Introductions

In order for introductions to be a viable data source for embedding a person, they must satisfy two requirements: (1) most people in a story should have at least one introduction (coverage), and (2) these introductions should encompass a variety of information (diversity).

3.2.1 Coverage requirement

We obtain 100 sample stories consisting of 10 to 100 text documents. For each story, all persons mentioned and their corresponding introductions are extracted. Across stories, 82% of persons have at least one introduction, with 10% having 5 introductions or more. Most people have at least one introduction, which can be used to build a Per2vec embedding. If in future more introductions are needed, co-reference resolution and analysis of other nearby sentences may be introduced.

3.2.2 Diversity requirement

The embeddings we want to produce are meant to represent the person, which means the information should be both context independent (job, status, family), and dependent (role in story, presence at event). We labeled 537 introductions for 100 random news articles. The results are shown in Table 2. In short, introductions can be varied, and the author deliberately selects these nuggets of information; in many cases they relate to the role of the person in the story.

4 Model and Training

We have proposed a set of patterns that extract introductions for each person. We now outline a method to build an embedding from the introductions to represent the person in context.

When representing a person, we have a preset idea of what information matters: professional occupation, nationality, level of education, etc. In each context, any one of these features might be more relevant than others, or some intersection (e.g., in the Harvey Weinstein scandal story, female movie star is a common trait of several people). We propose to train a neural network to read through the introductions of a person and predict a set of categories for that person. If we force the

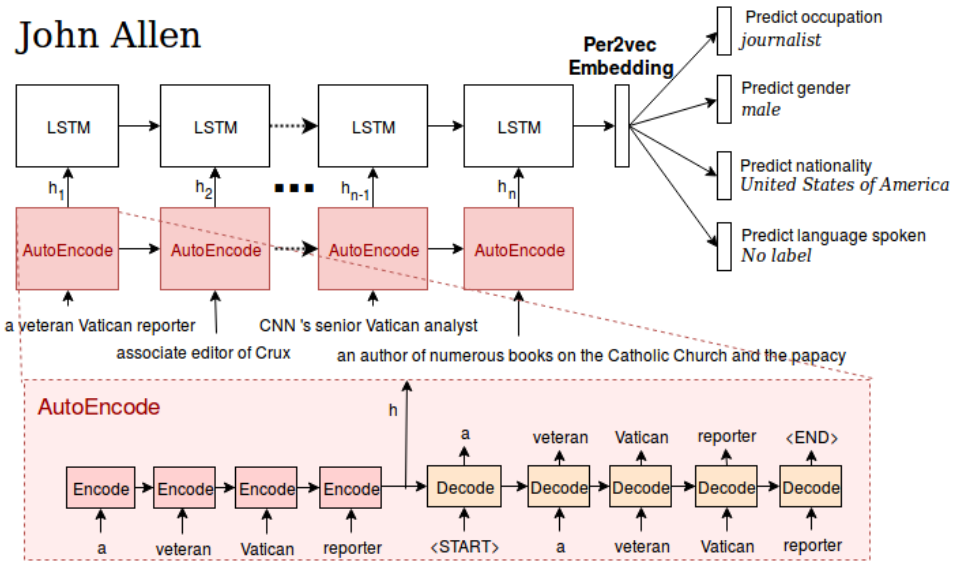

Figure 2: The model diagram: the main block is an LSTM, and the AutoEncoder is a module trained to learn to encode a sequence of words into a vector h. Up to 10 introductions from one story are used to create an embedding; shown here are introductions for the Vatican analyst John Allen.

neural network to make all its decisions based on a single vector, the neural network is forced to effectively fit any information it deems relevant into the vector; this enables the neural network to "embed" the person. The rest of this section details the model architecture, the dataset used for training, and the results obtained.

4.1 Model Architecture

Figure 2 outlines the neural network architecture to embed a person's introductions. The task is to take a list of introductions for a given person in a story, and predict a set of attributes for that person.

There are three main components to the model. The AutoEncoder component is fed a single introduction and produces a vector that represents this introduction. Each word is represented by its GloVe word representation (Pennington et al., 2014). The AutoEncoder has a loss based on the decoder: from the encoded vector of the introduction, the decoder has to recover the introduction. The embedding of each introduction is then fed to the main LSTM, which is effectively reading through the list of introductions, and encoding them into a final vector representation of the person: Per2vec. We use LSTM cells (Hochreiter and Schmidhuber, 1997) for both the AutoEncoder and the main encoder.

The final embedding for the person is used to predict 4 attributes: occupation, gender, nationality and languages spoken. This procedure enforces that the model includes all information useful to predict the 4 attributes into the person embedding. We chose these 4 attributes as they each encompass an important aspect of a person, and would serve to differentiate people in a news story. Other attributes could be added based on application, such as age, religion, etc. Each attribute prediction task has a loss, and the AutoEncoder has a loss; the model is trained on the sum of these losses. The AutoEncoder module can be pre-trained ahead of time to speed up training of the full model.

4.2 Dataset for Training

The network proposed above can be used in two settings: in production (when looking to embed people of new documents), and during training. In production, a sequence of introductions about any given person can be fed in to obtain the person's embedding. However, at training time, labeled attributes have to be provided. Our training dataset is of the form: (person, person's introductions, person's attributes).

Using a dataset which contains 6 million news articles, organized into coherent stories, we extracted all entities using spaCy's NER model, and kept the 1 million entities of the type PERSON with most mentions per story. Each person was then searched for in Wikidata, which returned a list of structured entries. The first result was chosen as the correct match. As explained in Section 5, choosing the first match will create some

incorrect links, but as we desire to train the neural network at scale, we tolerate the noise in the training data.

From the linked Wikidata entry we extract the person's entries for occupation (P106), gender (P21), citizenship (P27) and languages spoken (P1412), if available. If the person does not have an entry on Wikidata (no person with that name), or has none of the 4 attributes filled, it is removed from our dataset. The final dataset consists of 150,000 people, which is split into 90% training and 10% testing. Table 3 gives statistics about the people attributes obtained for the 150,000 people in our final dataset, as well as the accuracy of each prediction task after training.

After training, the model can be used in production and doesn't require linked attributes to embed a person. One useful outcome of the model structure is to extract attributes for the person, as long as a low accuracy is tolerated for some attributes (see Table 3). For instance, given a story, people who have the occupation politician can be identified. We note that the low accuracy at predicting attributes such as occupation is expected since the dataset contains 400 fine-grained occupations, and predicting an occupation from introductions is not always feasible.

To evaluate model choices, we perform two ablation studies. We use the varied person embeddings on a secondary task (Entity linking, described in Section 5), results are shown in Table 4. First we evaluate whether using all 4 attributes in the training process is necessary. Removing any attribute is detrimental to performance at entity linking, therefore each attribute enriches the person embedding. The second ablation study evaluates the effects of increasing the number of introductions as input to the embedding process. The results show that more introductions yield higher accuracy on downstream tasks, and therefore richer person embeddings.

Attribute type	**# labels**	**% filled in**	**% acc.**
Occupation	400	84.6	41.1
Gender	2	99.0	90.5
Nationality	150	81.9	58.7
Languages Spoken	100	18.9	85.7

Table 3: Training attribute statistics. *# labels* is the number of classes for the classification task, *%filled-in* is the percentage of people that have that attribute in Wikidata. *% accuracy* is the accuracy of the trained model on the test set.

Method	**Accuracy (%)**
Per2vec_glb	69.2
Per2vec_o	78.7
Per2vec_all	81.5
Per2vec_1	72.5
Per2vec_2	73.9
Per2vec_5	79.1
Per2vec_10	81.5

Table 4: Ablation study results. Accuracy on entity linking. Top: varying attributes included at training. glb: using gender, languages and nationality; o: using occupation only; all: using all 4 attributes. Bottom: comparison of results using at most x introductions.

5 Evaluation: Entity Linking

When extracting entities from text, it is common to link them to a knowledge base to gain access to a wealth of structured information. However, no knowledge base is complete, and a person mentioned in text can be missing from the knowledge base, in which case its link is *nil*.

Entity linking is particularly arduous when a name is common: a person might not be on Wikidata, but there might be tens people with identical names (homonyms), in which case the linking algorithm has to rebut each entry and output *nil*.

We use Wikidata, a large, open-source knowledge base to link to. Wikidata encompasses Wikipedia: any Wikipedia page about a person has a Wikidata entry (Vrandečić and Krötzsch, 2014).

When randomly sampling 5000 person mentions in news text, 36% of the names do not have an exact entry on Wikidata, 44% of the names have exactly 1 entry with that name, and 20% of the time, there are 2 entries with that name. This indeed proves to be a problem as shown below, where competitive systems like Hoffart et al. (2011)'s get high accuracy when a link exists (80%), but low accuracy (15%) when the correct link is NIL but a homonym exists in the KB.

5.1 Two-Stage Linking System

We propose a two-stage linking system, shown in Figure 3. The first stage (Nil Or Match) decides, before looking into the knowledge base, whether to output *nil*, or to attempt the second stage: Matching, which assigns a score to each homonym based on their description in the knowl-

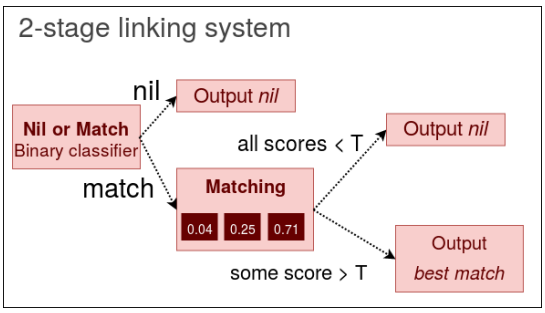

Figure 3: Two-stage stage linking system: Nil or Match followed by the Matching stage. This architecture helps improve accuracy at predicting *nil*.

edge base. If all matches in the second stage are below a threshold, the second stage outputs *nil*, otherwise the highest scoring homonym is linked.

Technically, each stage is implemented as a neural network whose input is the person embedding. The first stage (Nil or Match) is a 2-layer binary classification neural network. The second stage is an LSTM whose initial state is the person embedding, and the inputs to the LSTM are the words (embedded with GloVe) of the English description of the Wikidata entry. The final state of the LSTM is used to get the matching score with a fully connected layer.

5.2 A Challenge Dataset

We have constructed a challenge dataset: we randomly selected 1200 people with common names (two or more homonyms on Wikidata) and stored the context in which they appear (news articles). We manually annotated the ground truth link: in 35% of cases the link was *nil*, and otherwise the WikiData IDentifier (WDID) is the recorded link.

This dataset is particularly challenging as each person can link to *nil* or match at least 2 homonyms. Therefore, we expect lower accuracy than reports for other datasets (Ling et al., 2015).

5.3 Results

We compare 4 systems on the challenge dataset:

1. **Top choice baseline.** Wikidata has a measure of an entry's popularity based on pageviews of the related Wikipedia page. This baseline never outputs *nil* (0% nil accuracy) and always selects the most popular homonym.

2. **GloVe Sum baseline.** The two-stage linking system, but instead of using the person embeddings, we simply sum the GloVe word vectors of the introductions for the person. Sum of word vectors is a common text embedding technique (Conneau et al., 2017).

3. **Per2vec embeddings**. The two-stage linking system using the person embeddings.

4. **Ambiverse**. A popular, industrial linking system based on work presented in (Hoffart et al., 2011)[3]. We provide the Ambiverse system all news articles in the collection in which the person is mentioned.

Method	nil Acc.	Matching Acc.	Overall Acc.
Top choice	0.0	69.1	48.0
Hoffart et al.	15.1	**82.5**	53.5
Glove Sum	51.9	59.3	56.5
Per2vec	**89.5**	75.4	**81.5**

Table 5: Results for the two steps in entity linking: accuracy at predicting *nil*, which represents 35% of the data; matching accuracy for the other 65% of the data. Overall accuracy combines both.

Results are shown in Table 5. Three accuracy measures are given for each method: (1) *nil* tag accuracy: how often did the algorithm correctly assign *nil* when it is the true tag, (2) matching accuracy: when there is a link (not-nil), what percentage does the system get right, (3) overall accuracy of system (combining both). *nil* accuracy is only 15% for Hoffart et al. (2011): more than 6 out of 7 *nil* links were wrongly attributed to a Wikidata homonym. However, Hoffart et al. (2011)'s accuracy at linking when a link exists is high (82.5%) which corresponds to numbers reported in their paper (note also that we give their algorithm the entire text of documents whereas Per2vec is exposed only to the text of introductions). The two stage process using Per2vec embeddings achieves best overall accuracy, as it is able to recognize when a person is not present in the knowledge base. Recall that this dataset is artificially hard: each person has a common name, which is not usually the case in practice.

6 Case studies: News stories

6.1 Fatal Southwest Airlines Incident

On April 17th 2018, an incident occurred in a flying Southwest Airlines plane. We have collected

[3]Using the public API at: www.ambiverse.com/natural-language-understanding-api/

124 news articles from 15 major news sources about the incident. In these 124 news articles, 95 distinct people are mentioned. We manually labeled each person with a corresponding Wikidata entry or a *nil* tag, 80 (84%) are not on Wikidata (*nil*), and 16 of these 80 have at least one homonym on Wikidata. Using Per2vec and the two-stage linking system, we achieve an overall linking accuracy of 89%, while Hoffart et al. (2011)'s system gets an accuracy of 84%. These accuracies are higher than ones reported in Section 5 because in this real-world example, many of the links are straight-forward: 64 of the 95 (67%) people have no homonyms on Wikidata, making the only option viable the *nil* tag.

The discrepancy between the two systems is accounted for in the 16 people with *nil* links that have Wikidata homonyms. On these 16 people, Hoffart et al. (2011)'s system makes 10 erroneous links. For example, Marty Martinez (a passenger on the flight), is linked to a homonymous professional wrestler, and Kevin Michaels (a gas turbine engineer) is attributed to an American film director. Our system makes only 4 such mistakes.

Figure 4 presents a screenshot from the news story user interface showing the most frequently mentioned people in the story. When a link to Wikidata is found, the person's presentation is enriched by a photograph.

6.2 Solving the John Allen Problem

We've shown in Figure 1 that different John Allens appear in 118 different news stories; we built an embedding for each one and manually labeled each with a corresponding Wikidata entry or a *nil* tag. U.S. General John Allen is the most common homonym, appearing 55% of the time. We ran linking using (Hoffart et al., 2011)'s system, which got an overall accuracy of 75.4%.

We expect John Allen identities to form clusters in the embedding space, which can help determine in an unsupervised way the number of John Allen identities. We perform clustering on the embedding space, and measure *purity* of the clustering obtained: the accuracy obtained when each identity is assigned to the most frequent label in the cluster. To produce the clusters, we ran PCA over the embeddings, followed by spectral clustering, which identified 8 clusters, achieving a purity of 89%.

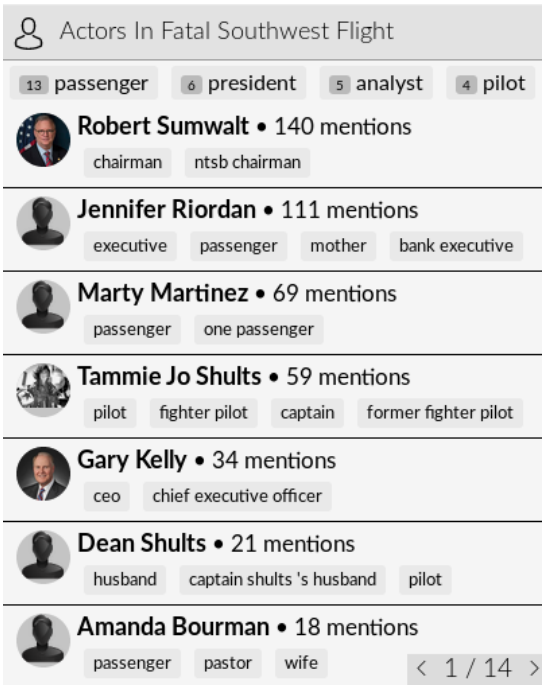

Figure 4: Most mentioned people in the Southwest Airlines Incident. People with a picture were linked to Wikidata. Role keywords extracted from introductions.

7 Conclusions and Future Work

We have presented a method to obtain continuous embeddings for people in context. We have proposed to extract introductions – targeted text about a person – from documents and base our embeddings solely on those. In order to train the model, we require a large dataset of people with annotated attributes, but when using the model at test time, no labels are required. We have shown person embeddings to be useful for entity linking, as they help predict whether the person is in a knowledge base or not, and achieve high overall accuracy on a challenging dataset released to the community. Finally, we've shown the embeddings used in practice for linking and disambiguation.

Future work includes using the person embeddings for subsequent tasks. One of the main uses of word vectors is as the input layer of a neural network for an NLP task: the neural network leverages the semantic information in the word embedding. Person embeddings could be used as well for person understanding. For instance, by extracting quotes for people in a story and training a model to learn to attribute quotes to people, effectively training a model to learn who said what. Person embeddings could also be used to discover person groups in a story and label person roles.

References

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *COLING-ACL*.

Razvan Bunescu and Marius Paşca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *11th conference of the European Chapter of the Association for Computational Linguistics*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Jinho D Choi and Andrew McCallum. 2013. Transition-based dependency parsing with selectional branching. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. volume 1, pages 1052–1062.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Marco Cornolti, Paolo Ferragina, and Massimiliano Ciaramita. 2013. A framework for benchmarking entity-annotation systems. In *WWW*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 462–471.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with wikipedia. *Artif. Intell.* 194:130–150.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 782–792.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear* .

Xiao Ling, Sameer Singh, and Daniel S. Weld. 2015. Design challenges for entity linking. *TACL* 3:315–328.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: linking documents to encyclopedic knowledge. In *CIKM*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)* 41(2):10.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *CoNLL*.

Ehsan Sherkat and Evangelos E Milios. 2017. Vector embedding of wikipedia concepts and entities. In *International Conference on Applications of Natural Language to Information Systems*. Springer, pages 418–428.

Masatoshi Suzuki, Koji Matsuda, Satoshi Sekine, Naoaki Okazaki, and Kentaro Inui. 2016. Fine-grained named entity classification with wikipedia article vectors. In *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*. IEEE, pages 483–486.

Andrew Trask, Phil Michalak, and John Liu. 2015. sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388* .

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. *arXiv preprint arXiv:1604.00425* .

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78–85.

Michael Wick, Pallika Kanani, and Adam Craig Pocock. 2016. Minimally-constrained multilingual embeddings via artificial code-switching. In *AAAI*. pages 2849–2855.