Distributed Representations for Semantic Matching in non-factoid Question Answering

Piero Molino*
University of Bari Aldo Moro
Bari, Italy

Luca Maria Aiello
Yahoo Labs
Barcelona, Spain

ABSTRACT

Users' interactions with search engines is shifting towards more complex information needs and the need for a deeper semantic understanding of the query intent is needed.

In this paper we propose a novel semantic matching criterion that adopts distributed representations of words in order to address complex information needs in a scalable way.

We show that combining this criterion with other well established features it is possible to obtain over 22% improvement for *MRR* and 27% in *P@1* over the best performing approach for answering non-factoid questions, a specific form of complex information need. Moreover we show that in our setting our criterion can substitute more complex linguistic feature.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.7 [**Artificial Intelligence**]: Natural Language Processing; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Question Answering, Learning to Rank, Semantic Matching, Natural Language Processing

1. INTRODUCTION

Search engines address individuals' information needs by retrieving documents that are relevant to the query and that are hosted in authoritative web pages. Documents are ranked by information retrieval techniques based on the query, the document and their proximity, but also based on the global structure of the connections between webpages and other factors. In the last years great efforts have been spent in integrating the capability of answering factoid questions in commercial search engines. Asking for the date of birth of an artist or the director of a movie results in a featured piece of information that aims to provide a specific, concise answer to the implicit user question. This sentence-level answers are obtained by means of entity type recognition and textual patterns [21].

The way users are interacting with search engines is also changing. Users take search capabilities for granted, and are becoming more demanding with respect to the ability of search engines to satisfy complex information needs [12]. Complex search missions are usually poorly represented by simple keyword queries; as a result, longer, more grammatically correct and syntactically structured queries are becoming increasingly frequent [4]. The trend is increased also by the gradual shift of the type of input device from keyboard towards speech, and on the reliance on query autocomplete functions.

The task of serving complex queries has been studied quite in depth by a conspicuous line of work on non-factoid Question Answering (QA), which have tried to address some of the main issues related to longer queries and complex information needs mainly in the form of *why*-questions [28] and manner questions [24]. The structure and the grammatical correctness of those questions make it possible to try to get a deeper understanding of the query intent. Even if non-factoid QA studies do not cover all the possible complex information needs, they are a good testbed for techniques that can also be applied at webscale for general purpose applications, so we focused on this subproblem, hoping tha the lesson learned will be useful for addressing the more general objective of answering webscale complex information needs.

Unlike other information retrieval tasks, in non-factoid Question Answering the linguistic analysis of both questions and candidate answers (usually short passages) has proven to be helpful for representing, matching and ranking [27].

As a result, a growing amount of linguistic features has been proposed and used in a learning-to-rank setting. That has allowed to obtain a very deep linguistic matching of questions and answers, but at a very high computational cost and with a considerable increase of the size of storage, mainly because of indexing of additional linguistic metadata annotations [3]. Also, given that answers are not long documents but usually short passages, synonymy and polysemy represent an additional problem that is usually not directly by linguistic features.

Continuous vector representations of words represent semantically similar terms with vectors that are close to each other in a multidimensional space, thus giving a more nuanced representation of the meaning of words also for short fragments of text, like questions

*This work was done while the first author was visiting Yahoo Labs Barcelona

SMIR 2014 Gold Coast, Australia

and answers [16].

In this work we address the problem of ranking candidate answers, proposing to use *distributed representations* learned with neural networks for matching questions and passages of text based on their semantic similarity. Our goal is twofold:

1. to deal with the shortness of the answers enriching their semantic representation to deal with synonymy and polysemy;
2. to show that features belonging to the family of linguistic analysis can be replaced, without loss of accuracy, with continuous vector representations that are smaller, faster to compute and more robust to noise.

Using a large public dataset from Yahoo Answers[1] we show that the adoption of a distributed semantic representation that is less computational expensive than linguistic analysis, balances the trade-off between accuracy and efficiency, improving the best baseline over 22% for *MRR* and 27% in *P@1*, while being more scalable.

The rest of the paper is organized as follows. We first describe the learning to rank framework for non-factoid Question Answering and the features adopted in Section 2. Next, in Section 3, we present the new adopted features based on distributed representations. Then, in Section 4, we present the experiments and the results of the proposed solution and compare them against the best solutions presented so far. Finally, we give a brief literature review in Section 5.

2. APPROACH

Our approach to the problem consists in three main steps: 1) representing both question and answers at different linguistic levels, 2) compute several different features that incorporate static properties of both answers and questions, like their length or their punctuation density, alongside with different measures of the similarity between them, 3) a ranking algorithm that exploits those feature to rank the answers according to a learned model.

As shown in Figure 1, during indexing time, snippets of text are analyzed by a Natural Language Processing (NLP) pipeline and the enriched representation is saved in an index. At search time, the questions are analyzed with the same pipeline, in order to obtain the same representation. The candidate answers are then obtained from the index and a feature extractor collects the feature from both question and answer representations. Those features are finally passed to a previously trained ranking algorithm that sorts the candidate answers according to its predictions.

Our NLP pipeline annotates both questions and passages with a stemmer, a part-of-speech tagger, a lemmatizer, a dependency parser, a named entity recognizer, a supersense tagger and a semantic role labeler.

2.1 Ranking Model

In the learning to rank approach, question-document pairs (q,d) are labeled with relevance judgments that indicate the degree of relevance of the document d with respect to query q. Each pair is represented by a set of features that are usually an indication of the degree of similarity between q and d, but also include information

[1]http://webscope.sandbox.yahoo.com

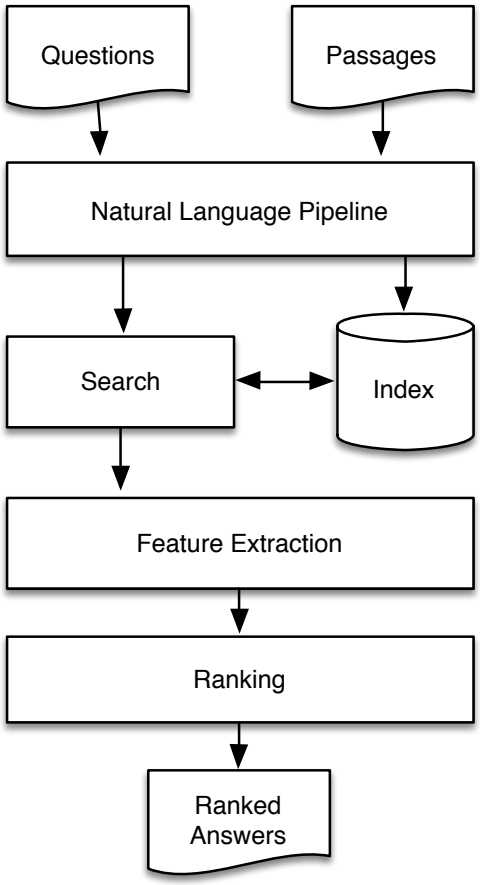


Figure 1: The overall architecture of the proposed approach. The NLP pipeline analyzes both questions and passages; linguistically annotated passages are stored in an index, wich is accessed by the search component with a query obtained from the linguistally annotated question; features are extracted from both candidate answers and questions and are used by the ranking component, that finally outputs a ranked answer list.

about q and d in isolation, such as their length or the PageRank of web documents. Each pair is treated as a single datapoint and a set of datapoints can be used for training purposes, in order to learn a function to predict the best ranking of different documents according to a query.

Several algorithm have been proposed for this goal in the literature [15]. We opted for Random Forests (RF) [9] because of its resilience to overfitting, a problem that may affect our experimental setting due to the size of our dataset, and because of the successful results in several use cases related to Community Question Answering [11] and in other large scale retrieval experiments [18].

Let $x_i = \phi(d,q)$, where ϕ is a feature extractor, and x_i is a m-dimensional vector. Let $D = (x_1,y_1),\ldots,(x_n,y_n)$ be a set of query-document pairs x_i and their associated relevance ratings $y_i \in Y$.

The RF algorithm trains a model H such that $H(x_i) \approx y_i$ and so that the ranking of all the documents d appearing in pair with a query q according to $H(x_i)$ is similar to the ranking according to y_i.

The algorithm is shown in Algorithm 1.

Algorithm 1 Random Forests

Require: $D = (x_1,y_1),\ldots,(x_n,y_n), r > 0$
1: **for** $i \leftarrow 1$ to r **do**
2: $D_t \leftarrow sample(D)$
3: $K \leftarrow roandomPick(m)$
4: $h_i \leftarrow buildDecisionTree(D_t,K)$
5: **end for**
6: $H() \leftarrow \frac{1}{r}\sum_{i=1}^{r} h_i()$
7: **return** $H()$

The main idea of RF is to apply a decision tree regression algorithm to M subsets of D and then average the results. A sample D_t is extracted with replacement from D (step 2). A set K of features is randomly picked from the feature set, so that $|K| \leq m$ (step 3). A decision tree is induced from D_t using the features in K (step 4). The whole process is repeated r times and the outputs of all the single trees are averaged to obtain the function H (step 6). The use of different samples of the data from the same distribution and of different sets of features for learning the individual decision trees prevent the overfitting.

In our experiments the queries are the questions and the documents are the candidate answers. In our experimental evaluation we adopted the implementation provided by the RankLib library[2] with the default parameters.

2.2 Features

In the following, we list a number of features that have been used in past work and that we use in combination with the feature we propose (3). The adopted features leverage the intuition that the similarity between the question and the answer and the intrinsic quality of the answer's text are good proxies for the quality of the answer itself. We divide the features in two different families: *text quality*, *linguistic similarity*. The first contains features already used in literature, the second mainly comes from non-factoid QA literature.

We also propose a completely novel semantic matching feature that exploits distributed representations of words, with a detailed description provided in section 3

2.2.1 Text Quality

Text quality features aim to estimate the intrinsic quality of an answer by capturing objective properties of the text composition. A summary follows.

Visual Properties. This group of features accounts for the visual appearance of the text composition. It includes the count of whitespaces and whitespace violations in the answer, the count of capital letters and capitalization violations, punctuation density, the number of URLs in the text, the quoted parts of the answer and so on. The number of capitalized words and the total count of punctuation marks are also counted, for a total of 24 features that are widely adopted in the literature [1, 11].

Readability. These features evaluate how easy is to read an answer. They consider the average word length in terms of number of characters and syllables and the ratio of complex words in the answer. Some readability indices are also adopted, such as Kincaid, Ari, Coleman-Liau, Flesch, Fog, Lix and Smog. These features were already adopted in literature [1, 11]. In total we used 15 readability features.

Informativeness. This group contains 3 simple features that count the amount of nouns, verbs and adjectives occurring in the answer but not in the question.

2.2.2 Linguistic Similarity

For the linguistic similarity features we followed an approach similar to that proposed by Surdeanu et al. [24]. We decided to adopt different levels of linguistic representation of text that can be obtained using NLP algorithms in order to construct tokens to be used by different similarity and overlap measures. For the NLP pipeline, we adopted the Snowball Stemmer[3], the ClearNLP suite[4], the SuperSense Tagger proposed by Ciaramita[5] and the CoreNLP named entity recognition[6]. This allows us to build representations of text using different lexicalization degrees: words, stems, lemmas, lemma and pos-tag concatenations, named entities and super senses as tokens. The representations are lists of token n-grams. As an example, the sentence "The man plays the piano", removing the stopwords, can be represented as word unigrams (man, plays, piano) or as lemma+pos unigrams (man:NN, play:VBZ, piano:NN) or as supersense bigrams (noun.person-verb.competition, verb.competition-noun.artifact).

We also tag the text with dependency parsing and semantic role labeling, so we can extract chains from them in the same way we extract the n-grams. For the dependency parsing the chains are constructed in the form of "dependant-relationType-head" but we can extract also more general chains that do not contain the relationType. For the semantic role labeling, the chain has the form of "predicate - argumentType - argument". Also in this case the argument type can be omitted. The length of the chain can be increased,

[2] `http://sourceforge.net/p/lemur/wiki/RankLib/`

[3] http://snowball.tartarus.org

[4] http://clearnlp.com

[5] http://sourceforge.net/projects/supersensetag/

[6] http://nlp.stanford.edu/software/corenlp.shtml

constructing longer chains concatenating the chains of length one that share intermediate elements. For example concatenating unlabeled dependencies from the previous example we can obtain the chains ("man - plays", "piano - plays").

Previous literature has shown how longer chains do not usually add valuable information because of their sparsity [24], so we do not adopt them. The tokens that compose the chain can also be at different lexicalization degrees, but in order to minimize the sparsity we adopted only lemmas and super senses. As for our example, from the sentence "The man plays the piano" we extract labeled dependencies lexicalized with lemmas ("piano - dobj - play", "man - nsubj - play"), their unlabeled versions ("piano - play", "man - play") and the versions with supersense lexicalization ("noun.artifact - dobj - verb.competition", "noun.person - nsubj - verb.competition") and ("noun.artifact - verb.competition", "noun.person - verb.competition"). The same is done with the semantic role labeling annotations, the possible chains are with argument labels with lemma lexicalization ("play - A0 - man", "play - A1 - piano"), without argument labels with lemma lexicalization ("play - man", "play - piano"), with argument labels and supersense lexicalization ("verb.competition - A0 - noun.person", "verb.competition - A1 - noun.artifact") and without argument labels with supersense lexicalization ("verb.competition - noun.person", "verb.competition - noun.artifact").

Overlap. The overlap features count the ratio of tokens in common between the question and the answer as $\frac{|t_q \cap t_a|}{|t_q|}$, where t_q is the set of tokens belonging to the question and t_a the set of tokens belonging to the answer.

With this simple overlap formula we calculate the overlap of unigrams at all the different lexical levels, resulting in 6 features. Other 15 features are obtained calculating the overlap of 2-grams, 3-grams and 4-grams of all the lexicalizations except the named entities, as they are already n-grams of words in most of the cases.

We also calculate the overlap of the dependency chains and semantic role labeling chains, both labeled and unlabeled and both with lemma and supersense lexicalizations, refulting in 8 features.

For the different lexicalizations of the unigrams we also calculate the Jaccard Index as $\frac{|t_q \cap t_a|}{|t_q \cup t_a|}$ resulting in additional 6 features. We do not calculate the Jaccard index for the n-grams and for the dependency and semantic role labeling chains because of their sparsity.

Frequency. We use standard information retrieval techniques to obtain a measure of similarity between question and answer that takes into account the frequency of the tokens in the texts and in the whole corpus. We assign scores to the question-answer pairs according to the Tf-Idf weighting scheme, to the BM25 weighting scheme and to the Language Modeling (with Dirichlet priors [30]) for all the different lexicalization levels except for the named entities, for a total of 15 features.

Density. We adopted a slight modification of the Minimal Span Weighting proposed by Monz [20], calculating it for all the different lexicalization levels. This gave us 6 features.

The original formula contains three components: a text similarity value, a span size ratio and a matching term ratio for balancing the local similarity value. As we have other features, like the frequency ones, that address for the text similarity, we retained only the local similarity part, resulting in the following formula:

$$\left(\frac{|\, t_q \cap t_a \,|}{1+\max(mms)-\min(mms)}\right)\left(\frac{|\, t_q \cap t_a \,|}{|\, q \,|}\right)$$

where t_q and t_a are the sets of tokens respectively of the question and the answer; $\max(mms)$ and $\min(mms)$ are the initial and final location of the shortest sequence of answer tokens containing all matching question tokens.

Machine Translation. As we mentioned earlier, machine translation approaches have been widely used in CQA and non-factoid Q&A. Their objective is to "bridge the lexical chasm" between the question and the answer. We calculate the probability of the question being a translation of the answer $P(Q \mid A)$ and use it as a feature:

$$P(Q \mid A) = \prod_{q \in Q} P(q \mid A)$$

$$P(q \mid A) = (1-\lambda) P_{ml}(q \mid A) + \lambda P_{ml}(q \mid C)$$

$$P_{ml}(q \mid A) = \sum_{a \in A} (T(q \mid a) P_{ml}(q \mid A))$$

where the probability that the question term q is generated from answer A, $P(q \mid A)$, is smoothed using the prior probability that the term q is generated from the entire collection of answers C, $P_{ml}(q \mid C)$ and is the smoothing parameter. $P_{ml}(q \mid C)$ is computed using the maximum likelihood estimator.

As the translation of a word to itself $P(w \mid w)$ is not guaranteed to be high, we set $P(w \mid w) = 0.5$ and re-scale $P(w'|w)$ for all the other w' terms in the vocabulary to sum up to 0.5, so that $\sum_{w' \in W} (w' \mid w) = 1$. As already pointed out in previous work [24], this is needed for the adoption of translation models for retrieval tasks, as the exact world overlap of question and answer is a good predictor.

Calculating the translation models for all the lexicalization degrees and for all the combinations of dependencies and semantic role labeling chains, we obtain 13 features.

Others. We consider 3 additional miscellaneous features: the length of the exact overlap of the sequences of words in the question and the answer normalized by the length of the question, the length ratio of the question and the answer, and the inverse of the length of the answer.

3. DISTRIBUTED REPRESENTATIONS OF WORDS

Continuous representations of words have been used to improve significantly many NLP applications [5, 10, 25]. In literature, several different models have been proposed for building these representations, among those the most widely used are Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA).

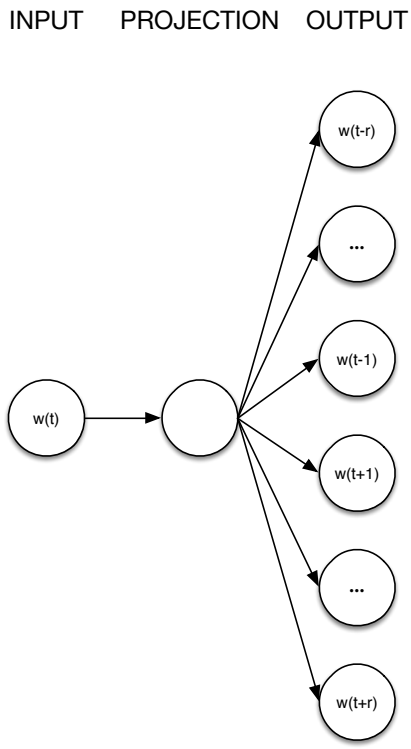


Figure 2: The architecture of the Continous Skip-gram Model.

In this paper, we adopt distributed representations of words learned by neural networks, because they have better performances than LSA in preserving linear regularities among words [17] and the latest models are computationally less expensive than LDA, so they scale better on large data sets. In [16], Mikolov et al. construct a very scalable log-linear classification network, using a simpler architecture than previous work, where neural networks are usually constructed with several non-linear hidden layers (e.g., [6]).

Two such simpler networks are proposed: the Continuous Bag-of-Words Model and Continuous Skip-gram Model. While both are shown to be effective in semantic-syntactic word relationship learning and sentence completion tasks, the former is faster to train, while the latter has better performances at the cost of slightly longer training time. Although both are really scalable, in our experiments we decided to adopt the latter one for its accuracy.

The Continuous Skip-gram Model builds on a Feedforward Neural Network described in [6], but it consists only of input, projection and output layers, so removing the hidden layer. As most of the complexity is caused by the non-linear hidden layer, this improves the learning efficiency at the expenses of a representation that might be less precise, but enables to learn models with bigger amounts of data. The model, shown in Figure 2, iterates over the words in the dataset and uses each word $w(t)$ as an input to a log-linear classifier with a continuous projection layer. What it outputs is a prediction of the words within a certain range before and after the input word.

As the words that are more distant from the input word are less related to it, the model adopts a randomization policy: if c is the fixed range before and after a word, a value r is obtained picking randomly a value between $[1,c]$. Then r words before the current and r words after the current are used as correct labels, from $w(t-r)$ to $w(t-1)$ and from $w(t+1)$ to $w(t+r)$.

At the end of the training phase, the weights associated with the projection layer are used as vector representations for each word. The resulting encoding captures meaningful word representations, where words of similar meaning have nearby representations.

In our case, we use word vector representations for building sentence level vector representation simply summing the vectors of the words that appear in the sentence. This way we obtain vector representations for questions and answers and we can compute their cosine similarity to obtain a semantic matching measure. We use this measure as one feature in the learning-to-rank setting.

The obtained vector represetation could be helpful for assessing synonymy problems, as words with similar meaning will occur in similar contexts resulting in similar vector representations. For instance the vectors for the word "cake" and for the word "pie" will be more close to each other than to the vector for the word "basketball". Consequently, vectors for sentences containing those words will be more similar, e.g. "John likes cakes" and "John likes pies" are more similar to each other thank to "John likes basketball".

4. EXPERIMENTS

In our experiment we aim to asses how the new proposed feature based on distributed representations perform in ranking answers in a Q&A portal. Given a question asked by a user the task cosists in predicting the best answer to that question ranking it the highest possible in the ranking of the candidate answers. This is a specific kind of complex information need and, even if is only a subproblem of webscale complex information needs, we believe that the lesson learned on this task by means of ranking algorithms and features can be useful also for the greater problem.

We compare the new proposed feature to more computationally expensive linguistic features and to text quality features. We also combine them in a single model, to see if the information they bring is sufficiently orthogonal to further improve the performances.

We measure the results by means of *Precision at 1* (*P@1*), and *Mean Reciprocal Rank* (*MRR*), an average sum of the inverse of the rank at which the correct answer is found, because we are interested both in finding the best answer in the first position and returning a ranking where the best answer is in the highest position possible.

When considering the answers to a single question, these are formally defined as follows:

$$MRR = \frac{1}{rank(BA)} \quad P@1 = rel_1$$

where $rank(BA)$ is the rank of the best answer for that question, and rel_i is an indicator function of relevance that returns 1 if the answer in the i^{th} position in the ranking is the best answer.

The distributed representations have been trained with text extracted from the Wikipedia dump of 3 march 2014, containing 3.5 million documents and 1.8 billion words. We decided to keep only words appearning in more than 5 documents, and trained the model with a window c of lenght 5, learning vector of 400 dimensions.

4.1 Dataset

Feature Set	P@1		MRR	
baseline	0.5091		0.6465	
d	0.6022	+18%	0.7652	+18%
l	0.618	+21%	0.7717	+19%
tq	0.6245	+22%	0.7857	+21%
d+l	0.618	+21%	0.7719	+19%
d+tq	**0.6476**	+27%	0.7907	+22%
l+tq	0.6401	+25%	0.7855	+21%
d+l+tq	**0.6476**	+27%	**0.7909**	+22%

Table 1: Experimental results. d distributed, l linguistic, tq text quality

The dataset we decided to use is Yahoo Webscope Manner Questions, an extraction of manner questions from U.S. Yahoo Answers data. The manner questions are extracted following two simple heuristics that aim at preserving only high quality questions and answers:

1. only questions that match the regular expression

   `how (to|do|did|does|can|would|could|should)`

   and have a correct answer selected by the asker or by the community are kept,

2. questions and answer with less than four words, out of which at least one is a noun and at least one is a verb, are filtered out.

This process produces 142,627 questions and 771,938 answers, with an average of 5.41 answers for each question. For each question there is a best answer selected by the asker or, if the asker did not select it, by the community of users by majority vote.

4.2 Performance Analysis

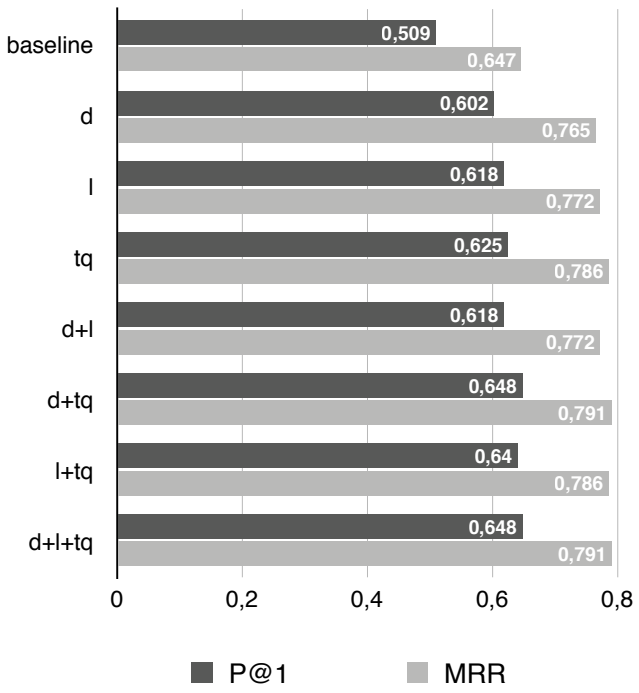
The features are split in the groups described in Section 2.2, where **l** refers to linguistic features and **tq** refers to text quality features. The group labeled with **d** contains only the feature based on the semantic matching of distributed representation.

A Random Forest model is learned for each feature set and the performances reported in Table 1 are the average of a 5-fold cross validation. In each fold 12.5% of the training set was used as a validation set.

The adopted baseline is, to the best of our knowledge, the best performance reported in literature on the same dataset [24]. It was obtained adopting a learning-to-rank setting similar to our, with a wide range of different features, including a subset of our linguistic features and web correlation based ones as well.

All the three groups improve over the baseline significantly both in *P@1* and *MRR*, with **tq** being the most effective. It is worth noticing that the distributed-representation based feature alone can compete with the other two groups of features, which are composed of 42 features for **tq** and 72 for **l**.

Taking into account the combinations of features we observe that the best performing one is the composition of **d** and **tq**. The combinations of **d** and **l** does not improve at all for *P@1* and improves



just of 0.002 for *MRR* over the **l** group alone, a non statistically significant improvement. This is expected as both groups try to intercept the topical similarity between question and answer.

The most interesting result that can be observed is that adding the **l** group to the previous best scoring group **d+tq** does not improve the performances at all for *P@1* and improves just of 0.002 for **MRR**, again a non statistically significant improvement. This finding suggests that in this setting the linguistic features, that requires a really expensive preprocessing time to be computed, can be substituted with a single features based on distributed representations of words without any loss of accuracy.

Finally, the best *P@1* scores obtained with the **d+tq** and **d+tq+l** feature groups are a **27%** improvement over the baseline, while the best *MRR* scores obtained with the **d+tq+l** features group are an improvement of **22%** over the baseline.

5. RELATED WORK

The approach we presented in this paper borrows insights from many different sources.

The question-to-answer transformation features based on translation models have been successfully applied in Question Answering before [7, 13, 22]. Recently, Matrix Factorization algorithms have been adopted for the same goal [31]. In out work, we include translation-based models in out feature set.

Linguistic representations of questions and answers have shown to be effective in matching questions to their best answers [26, 28]. We adopt the linguistic representation for computing some of the features, but we combine them with text quality features.

Severyn et al. [23] use a family of syntactic kernels to compute similarity between question and answers and rank answers accord-

ingly. The syntactic dependency features we use are a simplified version of those kernels taking into account only substructures of the whole kernel.

Bilotti et al. [8] label the semantic roles of the questions and match predicate-argument structures with the expected answer types, resulting in an improvement of the ranking. In our work, the predicate-argument features mimic this approach.

Learning translation models for different lexico-syntactic representations of text was proposed first by Surdeanu et al. [24]. In their work, they also use a learning framework for combining different families of features, focusing on linguistic ones, and adopt the supersenses for abstracting the lexical representation. We follow a similar approach, but we show that comparable results can be achieved with simpler features that rely on distributed representations.

Distributional semantics features have also been used for ranking in non-factoid Question Answering [19]. The adoption of Latent Semantic Indexing and Random Indexing in order to obtain vector representations of words and texts is shown to be valuable for ranking. In this paper, we follow the same approach, but we exploit a different algorithm that better preserves linguistic regularities [16] for learning the vector representations. Other lexical semantics solutions leverage Wikipedia entities [31], showing potential in addressing the retrieval of synonyms and hyperonyms. Last, Recurrent Neural Network Language Models [29] have been successfully explored in this context, confirming that lexical semantics is suitable to tackle the problem.

The task of best answer prediction on data extracted from social and collaborative Question Answering websites has been approached also through the assessment of the answer quality [1] and of the expertise of the answerer through network approaches [14, 2]. In general, exploiting both textual content and metadata have been considered [11]. We borrowed the idea that the intrinsic quality of the answer is a valuable indicator for ranking, but we were not able to use any metadata information, in particular about users, as that is not available in the Yahoo Webscope Manner Questions dataset, for privacy reasons.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presentad a semantic matching feature based on distributed representations of words. We have shown that combining this feature with text quality based ones in a learning to rank setting, it is possible to a chieve a 22% improvement for *MRR* and 27% in *P@1* over the best approach for answering non-factoid questions.

The feature based on distributed representations is more scalable that other linguistic features as it requires less preprocessing and a faster learning time, but in our settings in can substitute them without any loss in accuracy.

We still need to further investigate the role of the parameters of the neural network model for building the distributed representations (the window size and the number of dimensions) as optimizing them can lead to better representations and consequently better matching. At the same time more complex functions for the composition of the vector representations should be investigated.

7. REFERENCES

[1] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proceedings of the international conference on Web search and web data mining*, WSDM'08, pages 183–194, New York, NY, USA, 2008. ACM.

[2] C. Aslay, N. O'Hare, L. M. Aiello, and A. Jaimes. Competition-based networks for expert finding. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'13, pages 1033–1036, New York, NY, USA, 2013. ACM.

[3] G. Attardi, L. Atzori, and M. Simi. Index expansion for machine reading and question answering. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012.

[4] C. Barr, R. Jones, and M. Regelson. The linguistic structure of english web-search queries. In *EMNLP*, pages 1021–1030, 2008.

[5] P. Basile. Super-sense tagging using support vector machines and distributional features. In *EVALITA*, pages 176–185, 2011.

[6] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[7] A. L. Berger, R. Caruana, D. Cohn, D. Freitag, and V. O. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *SIGIR*, pages 192–199, 2000.

[8] M. W. Bilotti, J. L. Elsas, J. G. Carbonell, and E. Nyberg. Rank learning for factoid question answering with linguistic and semantic constraints. In *CIKM*, pages 459–468, 2010.

[9] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[10] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[11] D. H. Dalip, M. A. Gonçalves, M. Cristo, and P. Calado. Exploiting user feedback to learn to rank answers in q&a forums: a case study with stack overflow. In *SIGIR*, pages 543–552, 2013.

[12] D. Donato, F. Bonchi, T. Chi, and Y. Maarek. Do you want to take notes?: Identifying research missions in yahoo! search pad. In *Proceedings of the 19th International Conference on World Wide Web*, WWW'10, pages 321–330, New York, NY, USA, 2010. ACM.

[13] A. Echihabi and D. Marcu. A noisy-channel approach to question answering. In *ACL*, pages 16–23, 2003.

[14] P. Jurczyk and E. Agichtein. Discovering authorities in question answer communities by using link analysis. In *CIKM*, pages 919–922. ACM, 2007.

[15] T. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[17] T. Mikolov, W. tau Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013.

[18] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. In *Yahoo! Learning to Rank Challenge*, pages 77–89, 2011.

[19] P. Molino, P. Basile, A. Caputo, P. Lops, and G. Semeraro. Exploiting distributional semantic models in question answering. In *ICSC*, pages 146–153, 2012.

[20] C. Monz. Minimal span weighting retrieval for question answering. In R. Gaizauskas, M. Greenwood, and M. Hepple, editors, *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering*, pages 23–30, 2004.

[21] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 41–47, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[22] S. Riezler, A. Vasserman, I. Tsochantaridis, V. O. Mittal, and Y. Liu. Statistical machine translation for query expansion in answer retrieval. In *ACL*, 2007.

[23] A. Severyn and A. Moschitti. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*, pages 741–750, 2012.

[24] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Comput. Linguist.*, 37(2):351–383, June 2011.

[25] J. Turian, L. Ratinov, and Y. Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[26] S. Verberne, L. Boves, N. Oostdijk, and P.-A. Coppen. Using syntactic information for improving why-question answering. In *COLING*, pages 953–960, 2008.

[27] S. Verberne, L. Boves, N. Oostdijk, and P.-A. Coppen. What is not in the bag of words for *why*-qa? *Computational Linguistics*, 36(2):229–245, 2010.

[28] S. Verberne, H. van Halteren, D. Theijssen, S. Raaijmakers, and L. Boves. Learning to rank for *why*-question answering. *Inf. Retr.*, 14(2):107–132, 2011.

[29] W.-t. Yih, M.-W. Chang, C. Meek, and A. Pastusiak. Question answering using enhanced lexical semantic models. In *ACL*, pages 1744–1753, 2013.

[30] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *SIGIR*, pages 334–342. ACM, 2001.

[31] G. Zhou, Y. Liu, F. Liu, D. Zeng, and J. Zhao. Improving question retrieval in community question answering using world knowledge. In *IJCAI*, 2013.