RESEARCH

Biomedical Semantic Indexing using Dense Word Vectors in BioASQ

Aris Kosmopoulos[1,2*], Ion Androutsopoulos[1,3] and Georgios Paliouras[2]

[*]Correspondence: akosmo@iit.demokritos.gr
[1]Department of Informatics, Athens University of Economics and Business, Athens, Greece
[2]Institute of Informatics and Telecommunications, National Center for Scientific Research "Demokritos", Athens, Greece
Full list of author information is available at the end of the article

Abstract

Background:
Biomedical curators are often required to semantically index large numbers of biomedical articles, using hierarchically related labels (e.g., MeSH headings). Large scale hierarchical classification, a branch of machine learning, can facilitate this procedure, but the resulting automatic classifiers are often inefficient because of the very large dimensionality of the dominant bag-of-words representation of texts. Feature selection quickly harms the accuracy of the classifiers in this particular task, and dimensionality reduction transformations (e.g., PCA-based) usually cannot be efficiently applied to very large corpora.

Methods:
We examine the use of dense word vectors, also known as word embeddings, as an efficient method of dimensionality reduction that makes hierarchical text classification algorithms more scalable in biomedical semantic indexing, without being less effective than the usual bag-of-words representation. We consider several approaches for the transition from dense word vectors to dense vectors that represent entire texts, proposing the approach that we believe fits better this domain. We experiment with flat and hierarchically expanded K-nearest neighbor classifiers that employ dense vector representations of article abstracts, examining the effect of various parameters. We also present a high precision system that can be combined with the Medical Text Indexer (MTI) system of the US National Library of Medicine (NLM) to improve its performance.

Results:
Our experiments were performed on biomedical semantic indexing datasets of the BioASQ challenge. We show that dense word vectors can lead to very large dimensionality reduction (from millions of features to just 200), compared to the usual bag-of-words representation, reducing significantly the training and classification times of the classifiers, without degrading their effectiveness. We also present experiments on the combination of our high precision system with MTI, showing improvements in overall performance.

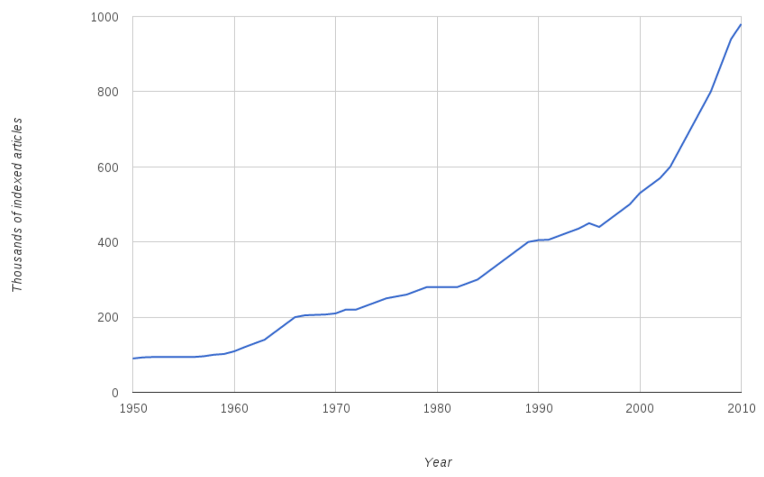
Conclusions: Dense word vectors (word embeddings) can lead to very large dimensionality reduction, making hierarchical text classification algorithms more scalable in biomedical semantic indexing, without degrading their effectiveness. K-nearest neighbor classifiers with dense vector representations of abstracts can help improve the performance of NLM's MTI semantic indexing system.

Keywords: Large Scale Hierarchical Text Classification; Semantic Indexing; Biomedical Curation; Continuous Space Word Vectors; Word Embeddings

Background

Curating biomedical articles is an overwhelming task because of the very large number of articles being published and the specialized knowledge required to understand and organize them. The online biomedical bibliographic database PubMed currently comprises approximately 24 million references and was growing at a rate of approximately 21,500 new articles *per week* in 2013. Figure 1 shows the number of biomedical articles indexed by PubMed per year.[1] A common curation task is to manually tag each newly published article with concepts from a controlled biomedical taxonomy, comprising tens of thousands of concepts. The tags and the taxonomy can then be used, for example, in search engines to retrieve articles whose concepts correspond to terms of the query (or their synonyms, hypernyms, etc.), or to hierarchically organize the retrieved articles [2].

Figure 1 Biomedical articles indexed by PubMed per year.

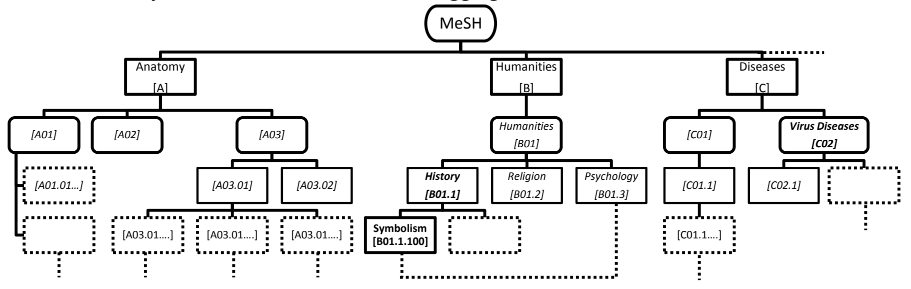


The US National Library of Medicine (NLM), the world's largest biomedical library, employs biomedical experts as curators to index biomedical journal articles by tagging them with *subject headings* (concepts) of the MeSH hierarchy [3].[2] Figure 2 shows a fraction of the MeSH hierarchy, which contains approximately 27,150 subject headings (shown as nodes). Tagging each article with one or more MeSH headings or, more generally, concepts from a taxonomy can be viewed as a problem of hierarchical classification [4]; in particular a case of multi-label hierarchical text classification [5]. The concepts are viewed as classes and the articles as instances to be classified; the term 'multi-label' indicates that there may be more than one correct concepts (classes) per instance (article) [6]. Specialized machine learning algorithms and evaluation measures have been proposed for hierarchical classification [7]. In Fig. 2, for example, mistakenly tagging an article with the heading *Symbolism* instead of *History* is less severe an error than tagging it with *Virus Diseases* and evaluation measures need to take such differences into account.

[1] All statistics obtained from [1] in March 2015.

[2] We consider only MeSH 'subject headings' (also known as 'descriptors'), not 'qualifiers' ('subheadings'), as in BioASQ.

Figure 2 A tiny fraction of the MeSH hierarchy. Assigning MeSH subject headings to each article is a case of multi-label hierarchical text classification. Tagging an article with *Symbolism* instead of *History* is less severe an error than tagging it with *Virus Diseases*.



Automatically assigning MeSH subject headings to biomedical articles is also one of the tasks of the annual Biomedical Semantic Indexing and Question Answering challenge (BioASQ, Task A) [8, 9]. In this task, a large corpus of abstracts (and titles) of biomedical journal articles (from PubMed) is provided to the challenge participants as training data, along with the MeSH subject headings the NLM curators assigned to the articles. During the challenge, the participating systems are given the abstracts (and titles) of newly published articles before the NLM curators have assigned them MeSH headings (for most journals, there is a curation backlog of a few weeks).[3] The systems report their decisions, and when the correct MeSH headings (of the curators) are available, the responses of the systems are evaluated. In practice, helping the curators assign subject headings in a semi-automated manner (e.g., by proposing possibly relevant headings) is a more realistic goal. The NLM has developed a system, the Medical Text Indexer (MTI), which is used for this purpose [10]. MTI follows a two-stage process to propose a set of MeSH subject headings for each article. In the first stage, the MetaMap tool [11] is used to identify concepts, ranks them and maps them to potential MeSH headings. In the second stage, a k-NN based approach identifies related articles and proposes extra MeSH headings based on them. MTI and its results are available to BioASQ participants as a baseline and a system to build upon.

Most hierarchical classification systems employ either *flat* classifiers or *cascades* of classifiers [5]. Flat classifiers [12, 13, 14] ignore the hierarchical relations between the classes (concepts), i.e., treat the classification problem as a non-hierarchical one, with as many (unrelated) classes as the number of nodes in the hierarchy. Cascades train a separate classifier for each node and usually operate in a top-down manner [15, 16]. Starting from the root, the classifier of each node decides if the instance (article) being classified should be passed on to any of the daughter nodes (and which ones), or if the instance should be classified as belonging in the current node. When classifying texts, both flat classifiers and cascades usually represent each text as a bag-of-words (BOW), i.e., a vector whose components (viewed as *features*)

[3]The PubMed identifiers of the articles are also provided. Hence, the participants may also consider the full text of the articles when it is available (e.g., via PubMed Central), but most (if not all) of the participating systems use only the abstracts.

show which words are present in the text (for Boolean features), or the TF-IDF (or other similar) scores of the words in the text, in both cases ignoring word order.[4]

Unfortunately, in large-scale hierarchical text classification BOW representations lead to millions of features, one for each vocabulary word (e.g., each word with at least a minimum number of occurrences in the training data) [18]. Standard feature selection techniques (e.g., Information Gain, χ^2 [19]) cannot be used to reduce significantly the number of BOW features, because in practice at least a few features (for characteristic vocabulary words) are needed per class, and there are thousands of classes (approximately 27,150 in our case). Classes at the upper levels of the hierarchy correspond hundreds or even thousands of descendant classes. Feature selection for such a class would be ineffective, since a lot of features (at least as many as the descendant classes of it) would need to be kept for classification. Indeed, the best performing flat classifiers of BioASQ use BOW features without feature selection [17]. In top-down cascades with BOW representations, feature selection at the top nodes of the hierarchy quickly degrades performance [20], because large numbers of features (vocabulary words) are needed to capture the very large topic variety of the texts those classifiers handle, and reducing the number of features quickly leads to indistinguishable instances (e.g., all-zero feature vectors) belonging in different daughter nodes. Methods like Principal Components Analysis (PCA) [21], which can be used to map the original vector space to a space of lower dimensionality (intuitively constructing new features for combinations of the original ones), cannot usually be applied to large scale hierarchical text classification, because of their computational complexity (e.g., PCA requires a costly eigen decomposition) [20].[5]

Another problem of BOW representations of texts is that closely words (e.g., singular and plural forms, synonyms) are treated as completely different (they are represented by different features). Stemmers are often used to group morphologically related words (e.g., convert 'proteins' to 'protein'), but standard stemmers perform poorly in biomedical texts [22]. Biomedical thesauri (e.g., UMLS [23]) and tools that identify and normalize biomedical terms (e.g., MetaMap [11]) can help group (e.g., represent by the same feature) closely related terms (e.g., synonyms), possibly increasing classification accuracy [24], but they do not decrease significantly the vocabulary size (number of BOW features). Very large numbers of features often lead to training and classification times that are prohibitively large for practical applications, even if implementations of machine learning algorithms optimized for (typically very sparse) BOW feature vectors are used.[6]

[4] TF-IDF scores are defined below. Features corresponding to n-grams [17] are much less frequently used in hierarchical text classification.
[5] In previous work [20], we proposed a PCA version efficient enough to be applied to the upper levels of a top-down cascade, but it decreased the cascade's accuracy. Furthermore, that PCA version is only applicable to tree concept hierarchies, but the MeSH hierarchy is not a tree (a node can have more than one parents).
[6] In our experiments, we use implementations optimized for each type of features. Large feature sets may also lead to over-fitting, but we do not study this issue here.

Methods

Dense word vectors

In BOW representations, each word of the vocabulary is in effect represented as a 'one-hot' vector with as many components (features) as the size of the vocabulary (millions of components in our case), and only one non-zero component (corresponding to the particular word). When Boolean features are used, the representation of a text is the sum of the vectors of its words, typically a very sparse vector (mostly zero components). In recent years, several methods have been proposed [25, 26, 27, 28, 29, 30] that map each vocabulary word to a *dense vector* (mostly non-zero components) of a space with a much lower dimensionality (often 100–300 dimensions in practical applications), so that words that occur in similar contexts in a large corpus are mapped to similar vectors (e.g., in terms of cosine similarity). Vectors of this kind, known as *continuous space word vectors* or *word embeddings*, have been found to capture morpho-syntactic and semantic properties of the corresponding words [31].

We show experimentally in the Results section that dense word vectors with as few as 200 features (components) allow performing large-scale biomedical semantic indexing as effectively as (and in some cases better than) with BOW vectors of millions of features, significantly reducing training and classification times. Previous work has already used dense vectors in flat text classification tasks (e.g., sentiment classification [26, 32]) and other biomedical text processing tasks (e.g., biomedical named entity recognition [33]), but not in hierarchical text classification and biomedical semantic indexing.

To obtain dense word vectors, we applied *word2vec* [26, 34] to the 10,876,004 English abstracts of biomedical articles that were provided as training data in the first year of the BioASQ semantic indexing challenge. We used the 'skip-gram' model of word2vec, which can be efficiently applied to very large corpora [26, 30, 35]. Roughly speaking, the model maps each vocabulary word w to a dense vector $\vec{w}$ that can be used to predict the (dense vectors of) the other vocabulary words w' that are likely to occur near w. We set the dimensionality of the dense vectors to 200, since previous work shows that 100–200 dimensions lead to reasonably good vectors [36].[7] Punctuation symbols, brackets etc. were removed and all letters were converted to lower case. All words appearing in fewer than five abstracts were ignored. The resulting dense vectors of 1,701,632 distinct words (the vocabulary) are available [37]. Constructing them took approximately 3 hours.[8]

As a preliminary investigation of the value of the constructed dense vectors, we identified the 100 most frequent words (excluding stop-words) of the 310 English biomedical questions that were provided in the first year of the BioASQ question

[7] We use the skip-gram model with negative sampling and default parameters, unless stated otherwise. For a detailed discussion of how word2vec relates to other methods that produce word embeddings and the broader field of distributional similarity, consult Levy et al. [30], who also report that the skip-gram model of word2vec is particularly efficient and robust in terms of performance.

[8] A server with an Intel Xeon 2.7 GHz CPU and 64 GB RAM was used. The word2vec implementation we used was single-threaded.

answering task (Task B).[9] For each of these 100 words, we selected its three closest words (among the 1,701,632), using the cosine similarity of the corresponding dense word vectors as a measure of word proximity. We then showed the 100 words and the closest three words of each one to two biomedical experts (members of the BioASQ biomedical experts group). For each of the 100 words, the two experts were asked to jointly mark the three closest words as relevant (closely related), possibly relevant, or irrelevant. Table 1 shows the 30 most frequent of the 100 words and the verdicts of the experts. Most of the closest words were found to be closely relevant. Notice that the closest words include several synonyms, semantically, and morphologically related terms (e.g., 'treatment'–'therapy', 'heart'–'cardiac', 'thyroid'–'thyroidal').

Table 1 Closest words to the 30 most frequent words of the BioASQ question answering task, using the cosine similarity of the dense vectors to measure proximity. Relevant (closely related) words are shown in bold, possibly relevant in normal font, and irrelevant (or misspelled) words in strikeout.

protein	**proteins**	**a-anchoring**	**pka-anchoring**
thyroid	**thyroidal**	**nonthyroid**	hyperfunctioning
associated	**correlated**	**related**	**correlates**
hormone	**gh**	**luetinizing**	**fshluteinizing**
human	murine	mouse	~~immortalized~~
used	**utilized**	**employed**	**applied**
genes	**gene**	**paralogs**	**operons**
treatment	**therapy**	**treatments**	**treating**
disease	**diseases**	disease-like	~~mmrn1rs6532197~~
gene	**genes**	**pseudogene**	**gene-encoding**
heart	**cardiac**	**chf**	**congestive**
role	**roles**	~~plays~~	~~play~~
affect	alter	modify	impair
dna	**dnas**	bisulfite-treated	polymerase-mediated
histone	**histones**	**h4k16**	**h4**
involved	**implicated**	**participates**	**regulating**
list	**lists**	**listing**	to-do
proteins	**protein**	**polypeptides**	**hsp70s**
known	~~yet~~	~~presently~~	well-known
patients	**outpatients**	**subjects**	whom
present	this	~~aimed~~	~~our~~
cancer	**cancers**	**crc**	~~caner~~
receptor	**receptors**	**hmc5**	5-nonyloxytryptamine
regulate	**modulate**	**regulates**	**orchestrate**
cell	**cells**	**cancer-cell**	**sw1710**
coding	**5-noncoding**	**5-untranslated**	**3-noncoding**
inhibitors	**inhibitor**	**small-molecule**	**atp-competing**
many	**several**	some	**numerous**
related	**linked**	**associated**	**relate**
cardiomyopathy	**cardiomyopathies**	**myocardiopathy**	**dcm**

Dense vectors for abstracts

Having computed the dense vectors of all the vocabulary words, the simplest method to obtain a dense vector $\vec{t}$ (of the same dimensionality) for a text $t = \langle w_1, w_2, \ldots, w_n \rangle$ of n consecutive word occurrences is to simply compute the *centroid* of the dense vectors $\vec{w}_i$ of the word occurrences:

$$\vec{t} = \frac{1}{n} \sum_{i=1}^{n} \vec{w}_i = \frac{\sum_{j=1}^{|V|} \vec{w}_j \cdot \mathrm{TF}(w_j, t)}{\sum_{j=1}^{|V|} \mathrm{TF}(w_j, t)} \tag{1}$$

[9]Task B uses datasets containing development and test questions, constructed by a team of biomedical experts, along with gold standard (reference) answers.

where $|V|$ is the number of (distinct) words in the vocabulary, and $\text{TF}(w_j, t)$ is the term frequency (number of occurrences) of the j-th vocabulary word in the text t.

Preliminary experiments that we performed indicated improved classification results by including the inverse document frequencies $\text{IDF}(w_j)$ in the centroids of the texts (in our case, abstracts), as follows:

$$\vec{t} = \frac{\sum_{j=1}^{|V|} \vec{w}_j \cdot \text{TF}(w_j, t) \cdot \text{IDF}(w_j)}{\sum_{j=1}^{|V|} \text{TF}(w_j, t) \cdot \text{IDF}(w_j)} \tag{2}$$

where $\text{IDF}(w_j) = \log \frac{|D|}{|D(w_j)|}$, $|D|$ is the total number of abstracts in the dataset we obtained dense word vectors from, and $|D(w_j)|$ is the number of those abstracts that contain the word w_j. In the remainder of this article, we use Eq. 2 to compute the centroids of abstracts. Having computed (off-line) the dense vectors of the vocabulary words and the IDF scores, computing the centroids of the approximately 4,000 (unseen) abstracts of a new weekly test set of the BioASQ semantic indexing task takes less than a minute.[10]

More elaborate methods to produce dense vectors for entire texts have also been proposed. Le and Mikolov [38] map not only vocabulary words, but also entire paragraphs (or sentences, or texts) to dense *paragraph vectors*, so that the dense vector of each paragraph can be used to predict the (dense vectors of) the paragraph's words; alternatively, the dense vector of each paragraph and the (dense vectors of) sequences of words from the paragraph are required to predict the (dense vector of) the words that follow the sequences. In both models, each paragraph vector can be thought of as a representation of the topics of the corresponding paragraph. Le and Mikolov [38] reported that the best paragraph vectors were the concatenations of the paragraph vectors of their two models. We used an implementation of their methods, called *sentence2vec* [39], to produce a paragraph vector for each abstract in our experiments, as an alternative to using abstract centroids.[11] The main limitation of sentence2vec is that it is very slow, given the scale of our datasets. Furthermore, it requires reprocessing the entire training dataset jointly with each new weekly test set of abstracts to produce paragraph vectors for the new abstracts. Dal et al. [36] compared paragraph vectors to LDA-based representations (representing each text as a distribution of latent topics) [41], BOW vectors, and centroids of dense vectors (without IDF scores). Their experiments showed that Paragraph Vectors in certain cases (e.g., measuring semantic similarity on Wikipedia articles) performed better than the other approaches, while in other cases they performed worse.

Large scale hierarchical text classification

As already noted, several hierarchical text classification systems use flat classifiers, in effect ignoring the hierarchy. In the Large Scale Hierarchical Text Classification

[10] On an Intel i7 3.2 GHz CPU with 32 GB RAM, using 8 threads.

[11] We note, however, that the documentation of sentence2vec is currently very limited, and it is unclear if it follows closely the methods of Le and Mikolov [38], and exactly which ones. An alternative implementation, called doc2vec, has recently become available [40], but it was not available during our experiments.

(LSHTC) challenges [42], for example, three of the best performing systems did not use the hierarchy at all [12, 13, 43]. Other methods use the hierarchy only to create features for flat classifiers [44], or use simplified forms of the hierarchy [15]. Among the systems that use the hierarchy (or a simplified form of it), top-down cascades are the most popular approach [15, 16]. They scale well to very large training datasets, compared to other hierarchy-aware classifiers [45, 46], but the higher-level classifiers propagate errors to the lower-level ones. Applying top-down cascades is also not straightforward when the hierarchy is not a tree (when a node can have multiple parents) and when texts belong in multiple classes (nodes).

In the BioASQ semantic indexing challenge, the hierarchy of MeSH headings is not a tree and each abstract belongs in 12 classes (nodes) on average. These factors make the successful use of the hierarchy more difficult, and most of the participants so far ignore it. The best system in the first year of the challenge, for example, used a flat Support Vector Machine (SVM) [17]. The most popular machine learning algorithms (used as flat classifiers or in cascades) are SVMs [47, 48], K-nearest neighbors (K-nn) [49], and (less frequently) Naive Bayes [50, 51]. BOW features are the most popular representation of texts, usually with TF-IDF values.

The Medical Text Indexer (MTI) [52] of NLM plays a central role in the BioASQ semantic indexing challenge, both as a baseline and as a component in several of the participating systems. The Default MTI is a high recall rule-based system which is available to all participants. MTI First-Line Indexer (MTIFL), on the other hand, is a high precision system, used to assign preliminary MeSH headings to journal abstracts; its decisions are then reviewed by expert curators. In BioASQ, the performance of MTIFL was lower than that of MTI.

Flat K-nn classifiers

In most of our experiments, we used flat K-nn classifiers.[12] During training, K-nn classifiers simply store the vector representations of the training instances (in our case, the BOW, dense centroid, or paragraph vectors of the training abstracts), along with their correct classes (the correct MeSH headings). In single-label classification, i.e., when each instance belongs in a single class, K-nn places each test instance (new abstract) in the class that is most frequent along the K training instances (neighbors) that are closest to the test instance, K being a parameter to be tuned. In BioASQ, however, classifying each test instance in the single, most frequent class of its K neighbors would miss many of the correct classes of the test instance. A simple alternative that we adopted was to consider each class separately, and classify the test instance in the class being considered if at least p (e.g., $p = 50\%$) of the K neighbors belong in the class. Since the main goal of our work was to demonstrate that dense vectors are overall a good option for large scale biomedical semantic indexing, we decided to ignore the problem of rare classes, but our experimental results confirm that rare classes need to be considered further.[13]

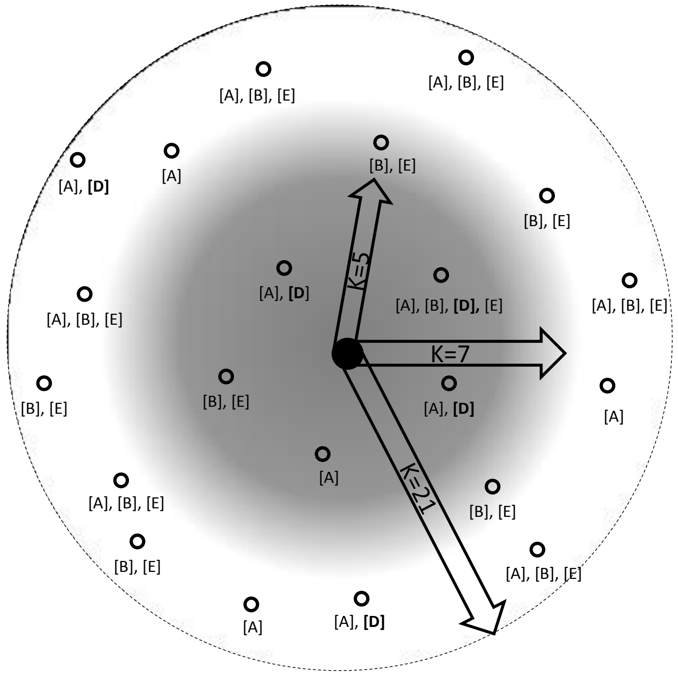
The value of K also greatly affects the behavior of the classifier. As K increases, precision improves, but recall decreases. Rare classes are particularly affected by

[12]We used our own K-nn implementation, optimized for BOW and dense vectors.
[13]Examples of rare classes are: *Streptogramin* and *Corpse Dismemberment*, with 5 and 4 instances, respetively, in the training dataset of the second year of BioASQ.

the value of K. In Fig. 3, for example, D is a relatively rare class, since it appears in only 5 of the 21 nearest training instances. Thus, for $K = 21$ it would not be selected if $p = 50\%$, but for $K = 5$ it would be selected (again if $p = 50\%$), since it would appear in 3 of the 5 nearest training instances. An interesting approach for future work would be to combine K-nn classifiers with different K values in a meta-classifier. The meta-classifier might learn to trust classifiers with small K values for rare classes, and classifiers with large K values for more frequent classes.

In large-scale classification, where there are tens of thousands of classes and millions of instances, K-nn is particularly popular, because its training time is minimal (at the expense of storing all the training instances), whereas other classifiers (e.g., SVMs) require much longer training times. On the other hand, K-nn is slower during prediction (when classifying test instances), since it has to compute the distance between the test instance and every (stored) training instance to obtain the K neighbors. Efficient implementations for sparse vectors (e.g., storing and comparing only non-zero features) can significantly reduce the prediction time of k-NN when BOW features are used. Dense vectors with small numbers of features also significantly reduce the prediction time. We used implementations optimized for each type of features in our experiments.[14] To compute the distances between training and test instances, we experimented with both Euclidean distance and cosine similarity.

Figure 3 Illustration of a K-nn classifier. A test instance is shown at the center, along with its K nearest training instances, for different values of K. The classes of the training instances are shown in square brackets.



Hierarchically expanded K-nn classifiers

We also experimented with a version of K-nn that exploits (still to a small extent) the class hierarchy. This version, which we call *hierarchically expanded K-nn* (HE-

[14]Further improvements are possible by indexing the training instances [53].

K-nn), treats the parents and children of a class C as being identical to C, when examining the K neighbors. Let us assume, for example, that $K = 5$, $p = 50\%$, that only 2 of the nearest 5 training instances belong in class C, and that another training instance among the 5 nearest ones belongs in a parent or child class of C. The simple K-nn would not classify the test instance in C, but HE-K-nn would. A second version of HE-K-nn, treats only the children (not the parents) of the class C as being identical to C.

High precision classification

Our experiments show that K-nn can provide precise predictions, especially for frequent classes. On the other hand, Default MTI (hereafter MTI) is a high recall system that performs well for rare classes. Therefore, we decided to combine the two systems. Since MTI already performs well, we aimed to increase its performance by adding to its predictions some classes for which K-nn is very confident, thus increasing MTI's recall further, without losing too much in terms of precision.

The first step in this hybrid MTI/K-nn approach is to select a relatively large value for K. A large K will cause K-nn to miss some of the rare classes, but we hope that MTI will compensate for this. The second step is to filter the predicted classes of K-nn using a separate logistic regression classifier for each class (node of the hierarchy).[15] We selected logistic regression, for this task, since it is scalable and does not require tuning (performs well with the default parameter settings). For each predicted class C of K-nn, the corresponding logistic regression classifier decides if the test instance will be classified in C or not. Filtering can be useful, in order to detect classes that were erroneously predicted by the K-nn, because of their high frequency in the datasets. The final step is to add the filtered K-nn predictions to the predictions of MTI.

More precisely, the logistic regression classifier of each class (node) C returns a probability estimate that the test instance belongs in C. If the estimate exceeds a threshold t, we classify the test instance in C. The logistic regression classifier of each class C was trained using the training instances of C as positive examples, and the training instances of the parents, children, and siblings of C as negative examples. When the negative examples were too few (less than 50), training instances from the parents and children of the siblings of C were added as negative examples.

Results

Datasets

All of our experiments were performed on datasets of the BioASQ semantic indexing task. To obtain dense word vectors, we applied word2vec to the 10,876,004 training abstracts of the first year of the task (Table 2). This dataset contains 18,560,735 distinct words, but we produced dense vectors only for words that occured in at least five abstracts of this dataset, which led to dense vectors for a vocabulary of 1,701,632 distinct words. The classifiers (K-nn, HE-K-nn, logistic regression) of our

[15]We used the Liblinear implementation [54] of logistic regression, with ridge regularization (L2), which performs better than lasso [55]. We used logistic regression (to filter the decisions of K-nn), because it is scalable to the size of the data and insensitive to its single parameter.

experiments were trained on the 4,499,338 abstracts of the training dataset of the second year of the BioASQ task; this dataset is smaller than the training dataset of the first year, because it comprises abstracts only from the particular journals the test abstracts of BioASQ come from. In experiments with BOW features, we used features corresponding to the 1,701,632 distinct words we had dense vectors for; we also experimented with BOW features corresponding to the 4,406,024 distinct words of the dataset we used to train the classifiers, but there was hardly any difference in the results. Recall, also, that in experiments with dense vectors, only 200 features are used. For testing, we used the test abstracts of the first five weeks of the second year of BioASQ (approx. 4,500 abstracts per week).[16]

Table 2 The datasets of our experiments. We applied word2vec to the training abstracts of the first year of the BioASQ semantic indexing task. The classifiers were trained on the training abstracts of the second year. For testing, we used the test abstracts of the first five weeks of the second year.

	Number of Abstracts	Distinct Words
Word2vec training dataset	10,876,004	18,560,735
Classifiers training dataset	4,499,338	4,406,024
Week1 test dataset	4,440	98,555
Week2 test dataset	4,721	100,994
Week3 test dataset	4,802	105,654
Week4 test dataset	3,579	84,623
Week5 test dataset	5,299	113,658

In experiments with sentence2vec, we used the 10,876,004 training abstracts of the first year to obtain dense word vectors (as with word2vec), and we then produced paragraph vectors for the 4,499,338 training abstracts of the second year and the 4,440 test abstracts of the first week. We did not experiment with test abstracts of the other four weeks using sentence2vec, mostly because it was very slow.

Evaluation measures

In hierarchical classification, two kinds of evaluation measures exist [7]. Flat evaluation measures ignore the hierarchy and treat all misclassification errors in the same way, regardless of the distance between the predicted and correct classes (Fig. 2). By contast, hierarchical evaluation measures take into account the positions of the predicted and correct classes in the hierarchy. The BioASQ semantic indexing task ranks the participating systems using both flat and hierarchical measures.

The main flat measure of BioASQ is micro-averaged F_1 (micro F_1), which is a combination (harmonic mean) of micro-averaged precision (micro precision) and micro-averaged recall (micro recall), but macro-averaged F_1 (macro F_1) is also reported. These measures are defined as follows.[17] The precision (P_i) of a class C_i (node of the hierarchy) is the number of test instances (abstracts) correctly classified in C_i (true positives, TP_i), divided by the number of test instances the classifier placed in C_i (true positives and false positives, $TP_i + FP_i$). The recall (R_i) of a class C_i is the number of test instances correctly classified in C_i (TP_i), divided by the number of test instances that truly belong in C_i (true positives and false negatives, $TP_i + FN_i$). The F_1 score of C_i combines P_i and R_i.

$$P_i = \frac{TP_i}{TP_i + FP_i}, \quad R_i = \frac{TP_i}{TP_i + FN_i}, \quad F_{1,i} = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i}$$

[16]These five test datasets are collectively known as 'Batch 1' of the second year.
[17]We compute all the measures using the 'oracle' service of BioASQ [56].

Macro precision and macro recall average P_i and R_i over all the $|C|$ classes.

$$\text{Macro Precision} = \frac{1}{|C|}\sum_{i=1}^{|C|} P_i, \quad \text{Macro Recall} = \frac{1}{|C|}\sum_{i=1}^{|C|} R_i$$

Macro F_1 combines macro precision and macro recall.

$$\text{Macro}\, F_1 = \frac{2 \cdot \text{Macro Precision} \cdot \text{Macro Recall}}{\text{Macro Precision} + \text{Macro Recall}}$$

By contrast, micro precision and micro recall are defined as follows.

$$\text{Micro Precision} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FP_i}, \quad \text{Micro Recall} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i + FN_i}$$

Micro F_1 combines micro precision and micro recall.

$$\text{Macro}\, F_1 = \frac{2 \cdot \text{Micro Precision} \cdot \text{Micro Recall}}{\text{Micro Precision} + \text{Micro Recall}}$$

The micro-averaged measures assign greater importance to classes with more test instances; consequently, rare classes do not influence much the results. By contrast, the macro-averaged measures assign equal importance to all classes. We also report accuracy scores, i.e., the number of correctly classified test instances divided by the total number of test instances. Accuracy is also a flat evaluation measure.

The main hierarchical measure of BioASQ is the Least Common Ancestors F_1 measure (LCA-F_1). This measure creates two graphs for each test instance, one for the predicted and one for the true classes of the instance. The graphs are created in a way that guarantees that they contain the minimum number of nodes each, and that their union graph contains a path that connects each predicted class to a correct class of the test instance and vice versa. LCA precision, LCA recall, and (their combination) LCA F_1 are then computed as operations between the sets of nodes of the two graphs. Consult [7] for a formal definition of this measure.

Experiments with flat K-nn classifiers

In a first set of experiments, we compared three flat K-nn classifiers that represent the abstracts as centroids (with TF-IDF weights) of dense word vectors, paragraph vectors, and BOW vectors (with TF-IDF values), respectively. We set $p = 50\%$ and $K = 5$ in all three classifiers. In all three classifiers, we used cosine similarity. We tested the classifiers only on the test dataset of the first week (Week 1 of Table 2), because some of these experiments were particularly time consuming.[18]

Table 3 shows the results of this first set of experiments. The centroids of dense word vectors led to better performance than BOW vectors in micro F_1 and LCA F_1 (BioASQ uses these two measures, to decide the winners of the challenge) and also in terms of accuracy. In contrast, BOW vectors were better in macro F_1. It

[18]This and all of the following experiments were performed using an Intel i7 3.2 GHz CPU (8 threads) with 32 GB RAM.

seems that BOW vectors represent rare classes better; hence, they win in macro F_1, which assigns equal importance to rare and frequent classes. Paragraph vectors also performed better than BOW vectors in terms of micro F_1, LCA F_1, and accuracy, and worse than BOW vectors in terms of macro F_1. Compared to dense vector centroids, paragraph vectors performed worse in all four measures (micro F_1, macro F_1, accuracy, LCA F_1). Also, the preprocessing time to construct the paragraph vectors of the test abstracts was prohibitively high (9,960 min), because sentence2vec preprocesses the training abstracts anew, along with the test abstracts, to produce paragraph vectors for the test abstracts. By contrast it took only one minute to produce the dense centroids or BOW representations of all the test abstracts. Most importantly, classifying all the abstracts of the first test week took only 50 minutes using the dense centroid or paragraph vectors (both use 200 features), but 470 minutes using the BOW representations (1,701,632 features).

Table 3 Comparing K-nn classifiers ($K = 5$, $p = 50\%$, cosine similarity) that use centroids of dense word vectors, paragraph vectors, and BOW vectors. The centroids of dense word vectors performed better than BOW vectors in Micro F_1 and LCA F_1, but worse in Macro F_1, indicating that dense word vector centroids represent frequent classes better. Paragraph vectors performed worse than centroids of dense vectors, and computing the paragraph vectors was particularly slow. Classifying abstracts using dense vectors was an order of magnitude faster than using the BOW vectors.

	Dense TF-IDF Centroids	Paragraph Vectors	TF-IDF Bag-of-words
Micro F_1	**40.8%**	39.9%	38.1%
Micro Precision	62.2%	64.6%	61.9%
Micro Recall	30.4%	28.9%	27.6%
Macro F_1	17.1%	13.3%	**20.1%**
Macro Precision	51.7%	52.8%	59.7%
Macro Recall	15.2%	11.8%	17.9%
Accuracy	**26.3%**	25.3%	24.6%
LCA-F_1	**34.5%**	32.5%	31.8%
LCA Precision	53.7%	54.4%	53.1%
LCA Recall	27.5%	25.3%	24.8%
Classification Time	**50 min**	**50 min**	470 min
Preprocessing Time	**1 min**	9,960 min	**1 min**

In further experiments, we measured the effectiveness of the K-nn classifiers that use dense vector centroids and paragraph vectors, now adopting Euclidean distance, rather than cosine similarity; we used the same test data (Week 1), as in the previous experiments. Cosine similarity is usually considered more appropriate than Euclidean distance for BOW vectors.[19] It is less clear if cosine similarity is also better for dense vectors (centroids of dense vectors and paragraph vectors). Hence we wanted to investigate in these experiments if the effectiveness of the two classifiers that use dense vector representations could be improved further by the use of Euclidean distance. Table 4 shows that Euclidean distance does not improve the overall performance of the classifier that uses dense vector centroids. The micro F_1, macro F_1, accuracy, and LCA-F_1 scores of this classifier were slightly higher when using cosine similarity and the precision (all three variants) of the classifier was slightly higher with the Euclidean distance. In the following experiments, we used the Euclidean distance in the classifier with dense vector centroids, because we aimed to eventually combine a high precision classifier with Default MTI. The overall performance of the classifier that uses paragraph vectors was also worse with

[19]Cosine similarity compares the directions of the vectors, intuitively their topical similarity, whereas Euclidean distance also considers their magnitutes.

Euclidean distance, but in this case the deterioration (in micro F_1, macro F_1, accuracy, and LCA-F_1) was surprisingly larger. We did not experiment any further with paragraph vectors and BOW features, given that the dense centroid vectors performed overall better and were computationally more efficient.

Table 4 Comparing K-nn classifiers ($K = 5$, $p = 50\%$) that use dense vector representations, with cosine similarity or Euclidean distance. Euclidean distance did not improve the overall performance of the classifier that uses dense vector centroids, but the difference (compared to cosine similarity) was small, and the precision of the classifier was slightly higher with Euclidean distance. The performance of the classifier that uses paragraph vectors was much worse with Euclidean distance.

	Dense TF-IDF Centroids		Paragraph Vectors	
	Cosine sim.	Euclidean dist.	Cosine sim.	Euclidean dist.
Micro F_1	**40.8%**	40.3%	39.9%	28.6%
Micro Precision	62.2%	*62.6%*	64.6%	66.6%
Micro Recall	30.4%	29.7%	28.9%	18.2%
Macro F_1	**17.1%**	15.9%	13.3%	5.7%
Macro Precision	51.7%	*52.2%*	52.8%	49.7%
Macro Recall	15.2%	13.9%	11.4%	4.5%
Accuracy	**26.3%**	25.8%	25.3%	16.9%
LCA-F_1	**34.5%**	33.6%	32.5%	23.3%
LCA Precision	53.7%	*53.9%*	54.4%	52.5%
LCA Recall	27.5%	26.6%	25.3%	16.4%

Table 5 shows the effect of K on the performance of the K-nn classifier that uses dense vector centroids (the best representation of our previous experiments) with Euclidean distance; we used the same test data (Week 1), as in the previous experiments. As the value of K increases, precision (all three variants) also increases, while recall (the corresponding variants) decreases. This is expected due to the requirement that at least half ($p = 50\%$) of the K nearest neighbors must belong in a class C_i, in order to classify the test instance into C_i.

Table 5 Performance of the K-nn classifier ($p = 50\%$) with dense vector centroids and Euclidean distance, for different values of K. Higher K values lead to higher precision at the expense of lower recall. In terms of overall performance (F_1 measures and accuracy), lower K values are better.

	$K = 5$	$K = 7$	$K = 13$	$K = 21$
Micro F_1	**40.3%**	39.7%	38.3%	37.3%
Micro Precision	62.6%	66.0%	70.3%	**72.4%**
Micro Recall	**29.7%**	28.4%	26.4%	25.1%
Macro F_1	**15.9%**	14.3%	11.6%	9.9%
Macro Precision	52.2%	56.7%	62.8%	**66.4%**
Macro Recall	**13.9%**	12.2%	9.8%	8.2%
Accuracy	**25.8%**	25.3%	24.2%	23.4%
LCA-F_1	**33.6%**	32.3%	30.4%	29.0%
LCA Precision	53.9%	55.1%	56.6%	**57.2%**
LCA Recall	**26.6%**	24.8%	22.4%	20.9%

Experiments with hierarchically expanded K-nn classifiers

We also experimented with our two hierarchically expanded versions of K-nn (HE K-nn), using Euclidean distance. Recall that the first expanded version (HE K-nn v1) treats the children and parents of a class C as identical to C. The second version (HE K-nn v2) treats only the children (not the parents) of C as identical to C.

Table 6 compares the performance of the two hierarchically expanded versions against a non-expanded K-nn classifier, for $K = 13$ (a mid-range value of Table 5), using dense vector centroids and Euclidean distance in all cases; we used the same test data as above (Week 1). As expected, the hierarchically expanded versions achieve higher recall at the cost of lower precision. HE K-nn v1 increases recall more than HE K-nn v2, since it expands each class being considered with more classes (both parents and children), at the cost of lower precision. HE K-nn v2 (the more conservative expanded version) had the best overall performance in hierarchical evaluation (LCA F_1), while HE K-nn v1 (the less conservative expanded version) had the best overall performace in flat evaluation (micro F_1, macro F_1, accuracy). The non-expanded K-nn had the best precision scores. Hence, in the following section, where we aimed to combine Default MTI with a high precision classifier, we used the non-expanded K-nn version.

Table 6 Comparing a flat K-nn classifier to two hierarchically expanded K-nn classifiers, using dense vector centroids, $K = 13$, $p = 50\%$, and Euclidean distance in all cases. HE K-nn v1 treats the children and parents of a class C being considered as identical to C. HE K-nn v2 treats only the children (not the parents) as identical to C. The hierarchically expanded versions achieve higher recall (especially HE K-nn v1) at the cost of lower precision. HE K-nn v2 had the best overall performance in hierarchical evaluation (LCA F_1), while HE K-nn v1 had the best overall performace in flat evaluation (micro F_1, macro F_1, accuracy). The non-expanded K-nn had the best precision scores.

	K-nn 13	HE K-nn 13 v1	HE K-nn 13 v2
Micro F_1	38.3%	**39.4%**	38.7%
Micro Precision	**70.3%**	44.1%	54.4%
Micro Recall	26.4%	**35.6%**	30.1%
Macro F_1	11.6%	**17.6%**	13.4%
Macro Precision	**62.8%**	34.3%	44.4%
Macro Recall	9.8%	**17.6%**	12.1%
Accuracy	24.2%	**24.5%**	24.3%
LCA-F_1	30.4%	33.7%	**34.3%**
LCA Precision	**56.6%**	48.0%	55.2%
LCA Recall	22.4%	**28.5%**	26.8%

Improving Default MTI

The reader is reminded that we aimed to improve the performance of Default MTI by adding high precision predictions of a K-nn classifier to the predictions of Default MTI, thus increasing its recall without losing much in terms of precision. Table 7 shows how the performance of Default MTI changes, when the predicted classes of our flat K-nn ($K = 5, 13, 21$, $p = 50\%$) with dense vector centroids and Euclidean distance are added to the predictions of Default MTI; again, we used the same test data (Week 1). We call the combined system MTI/K-nn; this system does not yet include the logistic regression classifiers that filter the predictions of K-nn. As expected, the best overall results were obtained for large K values ($K = 13$ and $K = 21$). MTI/21-nn was the best performer in terms of micro F_1 (the main non-hierarchical measure of BioASQ) and also in terms of accuracy, whereas MTI/13-nn was the best performer in terms of LCA-F_1 (the main hierarchical measure of BioASQ) and macro F_1, but the differences between MTI/13-nn and MTI/21-nn were very small. The overall improvements of MTI/13-nn and MTI/21-nn (compared to Default MTI) were due to their increased recall, which was obtained at the expense of a smaller decrease in precision. The larger the value of K, the smaller the decrease in precision, but also the smaller the increase in recall.

Table 8 shows the effect of adding the logistic regression classifiers (one per hierarchy node) to MTI/K-nn; we call the resulting system MTI/K-nn/LR. We considered only $K = 21$ in these experiments, since the overall performance of MTI/13-nn was very similar to that of MTI/21-nn in Table 7; we used the same test data as above (Week 1). Recall that in MTI/K-nn/LR, for each predicted class C of the K-nn classifier, the logistic regression classifier of C returns a probability estimate

Table 7 Adding the predictions of a flat K-nn classifier ($K = 5, 13, 21$, $p = 50\%$) that uses dense centroid vectors and Euclidean distance to the predictions of Default MTI. The best overall results (micro F_1, macro F_1, accuracy, LCA-F_1) were obtained using the combined MTI/K-nn system for large K values. The improved performance of MTI/K-nn (compared to Default MTI) was due to increased recall, at the expense of a smaller decrease in precision. The larger the value of K, the smaller the decrease in precision, but also the smaller the increase in recall.

	Default MTI	MTI/5-nn	MTI/13-nn	MTI/21-nn
Micro F_1	57.0%	56.7%	57.4%	**57.5%**
Micro precision	**58.3%**	53.8%	56.1%	56.5%
Micro recall	55.9%	**59.9%**	58.8%	58.5%
Macro F_1	48.1%	48.4%	**48.5%**	48.4%
Macro precision	**54.9%**	51.5%	53.9%	54.3%
Macro recall	51.7%	**53.2%**	52.4%	52.2%
Accuracy	40.8%	40.4%	41.1%	**41.2%**
LCA-F_1	48.4%	48.6%	**48.9%**	48.8%
LCA precision	51.5%	48.9%	50.3%	**50.5%**
LCA recall	48.6%	**51.5%**	50.7%	50.5%

that the test instance belongs in C. C is added to the predictions of Default MTI if the probability estimate exceeds a threshold t. We experimented with several thresholds, eventually setting $t = 0.5$ (in all classes), since the results were not particularly sensitive to the choice of t. Unfortunately, the overall effect of the logistic regression classifiers was minimal. As shown in Table 8, they increased the precision of the combined system, but at the expense of lower recall, and there was a very small or no difference in the overall scores (micro F_1, macro F_1, accuracy, LCA-F_1). Given the complexity of training a separate logistic regression classifier per class, MTI/K-nn seems a better choice than MTI/K-nn/LR for practical purposes.

Table 8 The effect of adding logistic regression classifiers (LR, one per node hierarchy) to filter the predictions of the K-nn classifier ($K = 21$, $p = 50\%$, dense centroid vectors, Euclidean distance) in the combined MTI/K-nn system. The overall effect of the logistic regression classifiers was minimal. Precision increased, but at the expense of lower recall, and there was a very small or no difference in the overall scores (micro F_1, macro F_1, accuracy, LCA-F_1).

	MTI/21-nn	MTI/21-nn/LR
Micro F_1	57.5%	**57.6%**
Micro Precision	56.5%	57.1%
Micro Recall	58.5%	58.1%
Macro F_1	**48.4%**	**48.4%**
Macro Precision	54.3%	54.6%
Macro Recall	52.2%	52.1%
Accuracy	41.2%	**41.3%**
LCA-F_1	**48.8%**	**48.8%**
LCA Precision	50.5%	50.7%
LCA Recall	50.5%	50.2%

Table 9 compares the performance of Default MTI and MTI/21-nn/LR.[20] The scores are now averaged over the five test datasets of Table 2 (Weeks 1–5), but the same conclusions hold for each individual week. The scores of the best performing system that participated in the BioASQ semantic indexing task in the same weeks (Weeks 1–5) are also shown, averaged over the five weeks.[21] The overall scores of MTI/21-nn/LR (micro F_1, macro F_1, accuracy, LCA-F_1) are better than those of Default MTI. In terms of LCA-F_1 (the main hierarchical measure of BioASQ), the score of MTI/21-nn/LR is also close to the score of the best system(s).

Interestingly, the best systems also seem to focus on frequent classes. Thus, although the 'Best System' has the highest scores in terms of micro F_1 and LCA-F_1

[20]The differences between MTI/21-nn/LR and MTI/21-nn were again minimal.
[21]The best performer was not the same system in all weeks.

Table 9 Comparing Default MTI to our combined MTI/21-nn/LR system and the winner of the BioASQ semantic indexing task (results averaged over Weeks 1–5 of Batch 1). The overall scores of MTI/21-nn/LR (micro F_1, macro F_1, accuracy, LCA-F_1) are better than those of Default MTI. In terms of LCA-F_1 (the main hierarchical measure of BioASQ), the score of MTI/21-nn/LR is close to the score of the winner. Also, the macro F_1 score of MTI/21-nn/LR was higher than the winner's, indicating that the winner focuses even more on frequent classes.

	Default MTI	MTI/21-nn/LR	Best System
Micro F_1	57.1%	57.7%	**59.0%**
Micro precision	58.5%	57.4%	59.4%
Micro recall	55.7%	58.1%	58.6%
Macro F_1	48.5%	**48.8%**	43.7%
Macro precision	54.7%	54.4%	59.8%
Macro recall	52.2%	52.4%	44.8%
Accuracy	40.8%	41.3%	**42.3%**
LCA-F_1	48.6%	49.0%	**49.5%**
LCA precision	51.8%	51.0%	51.5%
LCA recall	48.8%	50.3%	50.5%

(the main BioASQ mesures), its macro F_1 is the lowest of the three systems. Hence, the appropriate handling of rare classes remains an open issue and the BioASQ evaluation measures could be extended to highlight it. Given the positive results we obtained with dense word vectors, it would be interesting to study the effect they would have on the performance of other systems that perform well in BioASQ.

Conclusions

Using datasets from the BioASQ challenge, we demonstrated that dense word vectors (word embeddings) can lead to very large dimensionality reduction, compared to the usual bag-of-word vectors, making hierarchical text classification algorithms more scalable to biomedical semantic indexing. We experimented with flat K-nn classifiers and hierarchically expanded variants, representing article abstracts either as (TF-IDF weighted) centroids of dense word vectors, or directly as dense paragraph vectors. Our experimental results indicate that despite the very large dimensionality reduction (from millions of features to only 200), the centroids of dense word vectors lead to similar or even better performance, compared to bag-of-word vectors. Paragraph vectors did not perform better than the centroids of dense word vectors, and constructing paragraph vectors was very slow for practical applications (at least using the particular implementation that was available).

We also showed that adding the predicted classes of a high-precision flat K-nn classifier with dense vector centroids to the predictions of the Default MTI system can improve the overall performance of the latter, by increasing its recall without losing much in terms of precision. Using additional logistic regression classifiers (one per hierarchy node) to filter the predictions of K-nn in the combined system led to very minor improvements.

To the best of our knowledge, this article is the first one to consider dense word (and paragraph) vectors in hierarchical text classification and biomedical semantic indexing. It would be particularly interesting to use dense word vectors in order to improve the results of the best systems participating in the BioASQ semantic indexing task (e.g., systems that use SVMs or other learning algorithms, instead of k-nn). We believe that most of the participating methods can benefit from the use of dense word vectors; depending on the approach, the benefit could be an increase in performance, dimensionality reduction, or both. It would also be useful to experiment with different dimensionalities (instead of 200 dimensions) in the dense word vectors, and alternative methods to produce dense word vectors (e.g., GloVe [27], or the CBOW model of word2vec [35]) and dense sentence vectors [57].

Since our K-nn classifiers (and other methods) do not perform well with rare classes, another interesting future direction would be to develop solutions for rare classes in particular. One step in this direction could be to boost precision for specific classes, by tuning a threshold on the score produced by the k best examples of each category. Another direction would be to filter the predictions of Default MTI, instead of only adding classes to its predictions. Finally, more work is needed to exploit the concept hierarchy, since our hierarchically expanded K-nn methods (and most of the BioASQ participants) use the hierarchy to a very limited extent.

Competing interests
The authors declare that they have no competing interests.

Acknowledgements
We would like to thank the BioASQ consortium and the members of the BioASQ biomedical experts group for all their work on the datasets and the evaluation infrastructure. Special thanks to Yannis Almyrantis and Dimitris Polychronopoulos for their help during the preliminary assessment of the dense word vectors. We would also like to thank Ioannis Pavlopoulos for his contribution in the construction of the dense word vectors of the experiments.

Author details
[1]Department of Informatics, Athens University of Economics and Business, Athens, Greece. [2]Institute of Informatics and Telecommunications, National Center for Scientific Research “Demokritos”, Athens, Greece. [3]Institute for the Management of Information Systems (Digital Curation Unit) and Institute for Language and Speech Processing, Research Center “Athena”, Athens, Greece.

References

1. Medline Trend: Automated Yearly Statistics of PubMed Results for Any Query. http://dan.corlan.net/medline-trend.html
2. GoPubMed, Transinsight’s Semantic Search for Life Sciences. http://www.gopubmed.com/
3. Medical Subject Headings. http://www.nlm.nih.gov/mesh/
4. Silla Jr, C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery **22**(1-2), 31–72 (2011)
5. Kosmopoulos, A., Gaussier, E., Paliouras, G., Aseervatham, S.: The ECIR 2010 large scale hierarchical classification workshop. In: Proceedings of ACM SIGIR Forum, vol. 44, pp. 23–32 (2010)
6. Tsoumakas, G., Katakis, I.: Multi-label classification: an overview. Database Technologies: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications (2009)
7. Kosmopoulos, A., Partalas, I., Gaussier, E., Paliouras, G., Androutsopoulos, I.: Evaluation measures for hierarchical classification: a unified view and novel approaches. Data Mining and Knowledge Discovery, 1–46 (2014)
8. Tsatsaronis, G., Schroeder, M., Paliouras, G., Almirantis, Y., Androutsopoulos, I., Gaussier, E., Gallinari, P., Artieres, T., Alvers, M., Zschunke, M., Ngonga, A.: BioASQ: A challenge on large-scale biomedical semantic indexing and question answering. In: Proceedings of the AAAI Fall Symposium on Information Retrieval and Knowledge Discovery in Biomedical Text, Arlington, VA, USA, pp. 92–98 (2012)
9. The Challenge BioASQ. http://bioasq.org/
10. Indexing Initiative: Medical Text Indexer (MTI). http://ii.nlm.nih.gov/MTI/
11. MetaMap, A Tool For Recognizing UMLS Concepts in Text. http://metamap.nlm.nih.gov/
12. Madani, O., Huang, J.: Large-scale many-class prediction via flat techniques. In: Proceedings of Large-Scale Hierarchical Classification Workshop at ECIR, Milton Keynes, UK (2010)
13. Cissé, M., Artieres, T., Gallinari, P.: Learning efficient error correcting output codes for large hierarchical multi-class problems. In: Proceedings of Joint ECML/PKDD PASCAL Large-Scale Hierarchical Classification Workshop at ECML/PKDD, Athens, Greece, pp. 37–49 (2011)
14. Mao, Y., Wei, C.-H., Lu, Z.: NCBI at the 2014 BioASQ challenge task: large-scale biomedical semantic indexing and question answering. In: Proceedings of Question Answering Lab at CLEF, Sheffield, UK (2014)
15. Xue, G.-R., Xing, D., Yang, Q., Yu, Y.: Deep classification in large-scale text hierarchies. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Singapore, pp. 619–626 (2008)
16. Liu, T.-Y., Yang, Y., Wan, H., Zeng, H.-J., Chen, Z., Ma, W.-Y.: Support vector machines classification with a very large-scale taxonomy. ACM SIGKDD Explorations Newsletter **7**(1), 36–43 (2005)
17. Tsoumakas, G., Laliotis, M., Markantonatos, N., Vlahavas, I.P.: Large-scale semantic indexing of biomedical publications. In: Proceedings of the 1st Workshop on Biomedical Semantic Indexing Adn Question Answering at CLEF, Valencia, Spain (2013)
18. Yuan, G.-X., Ho, C.-H., Lin, C.-J.: Recent advances of large-scale linear classification. Proceedings of the IEEE **100**(9), 2584–2603 (2012)

19. Forman, G.: An extensive empirical study of feature selection metrics for text classification. The Journal of Machine Learning Research **3**, 1289–1305 (2003)
20. Kosmopoulos, A., Paliouras, G., Androutsopoulos, I.: The effect of dimensionality reduction on large scale hierarchical classification. In: Proceedings of the 5th Conference and Labs of the Evaluation Forum, Sheffield, UK, pp. 160–171 (2014)
21. Jolliffe, I.: Principal Component Analysis. Wiley Online Library, ??? (2002)
22. Han, B., Obradovic, Z., Hu, Z.-Z., Wu, C.H., Vucetic, S.: Substring selection for biomedical document classification. Bioinformatics **22**(17), 2136–2142 (2006)
23. Unified Medical Language System (UMLS). http://www.nlm.nih.gov/research/umls/
24. Zhu, D., Li, D., Carterette, B., Liu, H.: An incremental approach for MEDLINE MeSH indexing. In: Proceedings of the 1st Workshop on Biomedical Semantic Indexing Adn Question Answering at CLEF, Valencia, Spain (2013)
25. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. The Journal of Machine Learning Research **3**, 1137–1155 (2003)
26. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of 27th Conference on Neural Information Processing Systems, Harrah's LAke Tahoe, USA, pp. 3111–3119 (2013)
27. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, pp. 1532–1543 (2014)
28. Levy, O., Goldberg, Y.: Linguistic regularities in sparse and explicit word representations. In: Proceedings of the 18th Conference on Computational Natural Language Learning, Ann Arbor, Michigan, pp. 171–180 (2014)
29. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in Neural Information Processing Systems, Montreal, Canada, pp. 2177–2185 (2014)
30. Levy, O., Goldberg, Y., Dagan, I.: Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics **3**, 211–225 (2015)
31. Mikolov, T., Yih, W.-t., Zweig, G.: Linguistic regularities in continuous space word representations. In: Proceedings of 13th Conference of the North American Chapter of Association for Computational Linguistics, Atlanta, US, pp. 746–751 (2013)
32. Maas, A.L., Ng, A.Y.: A probabilistic model for semantic word vectors. In: Proceedings of Deep Learning and Unsupervised Feature Learning Workshop at NIPS, Whistler, Canada (2010)
33. Tang, B., Cao, H., Wang, X., Chen, Q., Xu, H.: Evaluating word representation features in biomedical named entity recognition tasks. BioMed research international **2014** (2014)
34. Word2vec, Tool for Computing Continuous Distributed Representations of Words. https://code.google.com/p/word2vec/
35. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of Workshop Track at ICLR, Scottsdale, USA (2013)
36. Dal, A.M., Olah, C., Le, Q.V., Corrado, G.S.: Document embedding with paragraph vectors. In: Proceedings of Deep Learning and Representation Learning Workshop at NIPS, Montreal, Canada (2014)
37. BioASQ Releases Continuous Space Word Vectors Obtained by Applying Word2Vec to PubMed Abstracts. http://participants-area.bioasq.org/info/BioASQword2vec/
38. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning, Beijing, China, pp. 1188–1196 (2014)
39. Sentence2vec, Tools for Mapping a Sentence with Arbitary Length to Vector Space. https://github.com/klb3713/sentence2vec
40. Doc2vec Tutorial. http://radimrehurek.com/2014/12/doc2vec-tutorial/
41. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. The Journal of machine Learning research **3**, 993–1022 (2003)
42. Large Scale Hierarchical Text Classification Challenge. http://lshtc.iit.demokritos.gr/
43. Puurula, A., Bifet, A.: Ensembles of sparse multinomial classifiers for scalable text classification. In: Proceedings of Large-Scale Hierarchical Classification Workshop at ECML/PKDD-PASCAL, Bristol, UK (2012)
44. Han, X., Liu, J., Shen, Z., Miao, C.: An optimized k-nearest neighbor algorithm for large scale hierarchical text classification. In: Proceedings of Joint ECML/PKDD PASCAL Large-Scale Hierarchical Classification Workshop at ECML/PKDD, Athens, Greece, pp. 2–12 (2011)
45. McCallum, A., Rosenfeld, R., Mitchell, T.M., Ng, A.Y.: Improving text classification by shrinkage in a hierarchy of classes. In: Proceedings of the 15th International Conference on Machine Learning, Madison, USA, pp. 359–367 (1998)
46. Cai, L., Hofmann, T.: Hierarchical document categorization with support vector machines. In: Proceedings of the 13th ACM International Conference on Information and Knowledge Management, Washington DC, USA, pp. 78–87 (2004)
47. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**(3), 273–297 (1995)
48. Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Springer, ??? (1998)
49. Jiang, L., Cai, Z., Wang, D., Jiang, S.: Survey of improving K-nearest-neighbor for classification. In: Proceedings of the 4th Conference on Fuzzy Systems and Knowledge Discovery, Haikou, China, pp. 679–683 (2007)
50. McCallum, A., Nigam, K., *et al.*: A comparison of event models for naive bayes text classification. In: Proceedings of Workshop on Learning for Text Categorization at AAAI, vol. 752, pp. 41–48 (1998)
51. Metsis, V., Androutsopoulos, I., Paliouras, G.: Spam filtering with Naive Bayes-which Naive Bayes? In: Proceedings of 3rd Conference on Email and Anti-Spam, Mountain View, USA, pp. 27–28 (2006)
52. Mork, J.G., Jimeno-Yepes, A., Aronson, A.R.: The NLM medical text indexer system for indexing biomedical literature. In: Proceedings of the 1st Workshop on Biomedical Semantic Indexing Adn Question Answering at CLEF, Valencia, Spain (2013)

53. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. In: Proceedings of the 4th Conference on Computer Vision Theory and Applications, Lisboa, Portugal (2009)
54. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J.: LIBLINEAR: a library for large linear classification. The Journal of Machine Learning Research **9**, 1871–1874 (2008)
55. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 267–288 (1996)
56. Oracle of BioASQ Participants Area. http://participants-area.bioasq.org/oracle/
57. Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA, pp. 1631–1642 (2013)