Learning to Identify Complementary Products from DBpedia

Victor Anthony Arrascue Ayala Trong-Nghia Cheng Anas Alzoghbi
Georg Lausen

Department of Computer Science
University of Freiburg
Georges-Köhler-Allee 051, 79110 Freiburg, Germany
arrascue|chengt|alzoghba|lausen@informatik.uni-freiburg.de

ABSTRACT

Identifying the *complementary* relationship between products, like a cartridge to a printer, is a very useful technique to provide recommendations. These are typically purchased together or within a short time frame and thus online retailers benefit from it. Existing approaches rely heavily on transactions and therefore they suffer from: (1) the cold start problem for new products; (2) the inability to produce good results for infrequently bought products; (3) the inability to explain why two products are complementary.

We propose a framework that aims at alleviating these problems by exploiting a knowledge graph (DBpedia) in addition to products' available information such as titles, descriptions and categories rather than transactions. Our problem is modeled as a classification task on a set of product pairs. Our starting point is the semantic paths in the knowledge graph linking between product attributes, from which we model product features. Then, having a labeled set of product pairs we learn a model; and finally, we use this model to predict complementary products for an unseen set of products. Our experiments on a real world dataset from Amazon show high performance of our framework in predicting whether one product is complementary to another one.

Keywords

Recommender Systems, Complementary Products, DBpedia, Knowledge-graph, Supervised Learning, Cross-domain.

1. INTRODUCTION

Online retailers like Amazon or eBay sell millions of products from several categories and continuously expand their catalogs. Being aware that browsing a large catalog can be overwhelming for a user and can therefore have a negative impact on revenue [3, 24], it is not a surprise that many of these retailers are equipped with Recommender Systems (RS) [28]. While recommending products based on the user's taste is certainly an important task any RS has to carry out, special attention has to be paid at the moment in which a purchase is about to be made. In this circumstance the RS can expose the user interested in a product to other products whose use is related to their desired product. Those products are referred to as *complementary* products [2]. For instance, a user who placed a guitar in the shopping cart might also be interested in purchasing a guitar tuner, a case, or a book about learning how to play it. It is important to notice that complementary products might belong to a different category than that of the associated product, e.g. *Instrument Accessories*, *Bags & Cases* and *Books* in the example above. This means that there are also some relationships at the level of categories too. Some of them are evident, like *Instruments* and *Instrument Accessories*, while some of them are not immediately obvious like *Instruments* and *Books*.

The majority of the approaches to find complementary products make use of transactional data, e.g. by extracting association rules or by applying other data mining techniques [7, 22, 27, 31, 32]. This means that new catalog products have no chance to appear as complementary to any other product. In addition to this, these approaches might also produce noisy recommendations for products which are infrequently purchased and are not able to explain the reason for the recommendation [18].

Therefore, a model is required from which this relationship between a complementary product and its associated product can be learned. We believe that a graph representation is the most suitable for this. Products, their attributes and the categories to which they belong, as well as the interconnections between them can be represented as nodes and paths in the graph. Having such a model would not only allow us to find links between a complementary product and its associated product without any need for transactions, but also to provide explanations of why this relationship exists.

However, learning to discriminate between different kinds of relationships from those interconnections within a complex semantic graph model is still an open problem. See for instance the feature vector illustrated in Figure 1, which represents a pair of products (p_i, p_j). Each of the features should indeed reflect a property which characterizes both products as a whole, or a certain interaction between those. A class label, *complementary* or *non-complementary* is associated with each sample. While trying to fit a classification model which learns how to discriminate the relationship from a set of samples like those might seem the obvious way of solving the problem, the problem remains: what are those features that reflect some property of *pairs of prod-*

Copyrights held by author/owner(s).
WWW2017 Workshop: Linked Data on the Web (LDOW2017)

CEUR-WS.org/Vol-1809/article-07.pdf

ucts? What do they reflect? Is this property related to the fact that one product might be complementary to the other one? This work aims at addressing these questions.

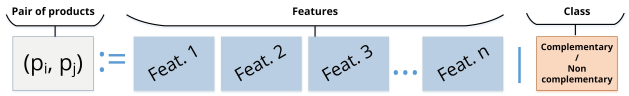

Figure 1: Feature vector

Contributions. The contributions of this paper can be summarized as follows.

1. We propose *CompLoD*, a novel and extensible framework that predicts the complementary relationship between two products by using only textual information such as titles, descriptions and categories.

2. The approach does not require transactions as other approaches and therefore, it is immune to the cold-start problem for new items.

3. Our scheme bridges the gap of how to encode information from a knowledge graph into a relevant feature set to fit a classification model and predict the complementary relationship.

4. Rather than building a knowledge graph from scratch, we believe that Semantic Web standards have enough to offer in this regard. Therefore, we use DBpedia[1], a well-known Linked Open Data (LoD) dataset, as the core of the knowledge graph. This dataset is the structured counterpart of Wikipedia and it is modeled as an RDF graph. We thereby demonstrate the usefulness of Semantic Web data.

We evaluate the accuracy of our approach by conducting experiments using Amazon data [14, 15] as our ground truth and find that it produces competitive results in comparison to those showed in [14] when taking into account that no transaction data is used. Our paper is structured as follows: We aim at providing some notation and definitions used within this paper in section 2. In section 3 the workflow of *CompLoD* is explained in detail. Section 4 shows the experiments that have been carried out to validate our system. In section 5 we present the related work. Finally, we present our conclusions in section 6.

2. NOTATION

This section provides some fundamental definitions which allow us to explain the system in detail. Our approach requires modeling four elements within an *RDF graph*: products, attributes and the categories to which products belong in addition to the relationships between all of these elements. Attributes are characteristics of a product which are extracted from a product's specification. Categories are instead given and are classified in a taxonomy. For instance, *bronze strings* is an attribute of a *guitar*. Moreover a guitar might belong to categories such as *Musical instruments*, *Guitars* and *Acoustic Guitars*. Categories are hierarchically arranged: acoustic guitars are a certain kind of guitar and these are a certain kind of musical instrument.

Since RDF is used as an underlying data model, we need to introduce some notation. Let I, B and L be the set of all IRIs (*Internationalized Resource Identifiers*), the set of all blank nodes and the set of all literals, respectively. Then a triple $(r_s, p, r_o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an *RDF triple*. An *RDF graph* G is a set of *RDF triples*. Moreover, let V be a set of variables. A tuple of the set $(I \cup L \cup V) \times (I \cup V) \times (I \cup L \cup V)$, e.g. $(?x, p, r_o)$, is called a triple pattern, in which variable names are prefixed with the question mark ("?") symbol. A triple pattern can be evaluated according to the SPARQL semantics to find all those mappings between variables and resources from the graph which would make of that tuple a valid triple. For a more comprehensive description of RDF and SPARQL we refer the reader to [13, 25].

A path in graph G between a source resource r_s and a target resource r_o is denoted by $\rho(r_s, r_o)_{(p_1,p_2,...,p_t)}$, where $(p_1, p_2, ..., p_t)$ is the sequence of predicates that connects r_s to r_o[2]. A path can be recursively defined as follow:

- $\rho(r_s, r_o)_{(p)}$ is a path of length 1 if there exists a triple $(r_s, p, r_o) \in G$ which connects r_s to r_o.

- $\rho(r_s, r_o)_{(p_1,p_2,...,p_t)}$ is a path of length t if there exists a resource r_l such that $\rho(r_s, r_l)_{(p_1,p_2,...,p_{t-1})}$ is a path of length $t-1$ and $\rho(r_l, r_o)_{(p_t)}$ is a path of length 1.

In our RDF graph each of the above-mentioned elements, products, attributes and categories, corresponds to an RDF resource. Let $P \subseteq I = \{r_{p_1}, r_{p_2}, ..., r_{p_n}\}$ be a subset of all IRIs representing a set of products, $A \subseteq I = \{r_{a_1}, r_{a_2}, ..., r_{a_m}\}$ is a set of products' attributes and $C \subseteq I = \{r_{c_1}, r_{c_2}, ..., r_{c_l}\}$ a set of products' categories. From now on products, attributes and categories will denote the RDF resource which identify these elements. Moreover, we will use r_p, r_a and r_c to denote an arbitrary product, attribute and category, respectively. We assume that each product is connected to its attributes and to the categories to which it belongs in G. Moreover, there exist arbitrary connections in G among attributes or categories, i.e. there are no restrictions on how resources are connected.

Problem statement. Given two products (r_{p_i}, r_{p_j}) in G, we would like to predict whether r_{p_j} is complementary to r_{p_i} using G. Let $comp(r_{p_i}, r_{p_j})$ be a function that returns 1 when this relationship holds, 0 otherwise. It is directional, i.e. $comp(r_{p_i}, r_{p_j}) = 1$ doesn't imply that $comp(r_{p_j}, r_{p_i})$ also returns 1[3].

3. OUR APPROACH

Figure 2 illustrates a high-level view of our framework *CompLoD*. The system needs to process only textual information of products, such as titles, descriptions and categories. Example 1 illustrates the input expected for a product.

[1]http://wiki.DBpedia.org/

[2]For simplicity, we assume $(p_1, p_2, ..., p_t)$ identifies a unique sequence of predicates which connect r_s to r_o.

[3]For instance, a guitar tuner is complementary to a guitar, but the opposite does not necessarily hold true.

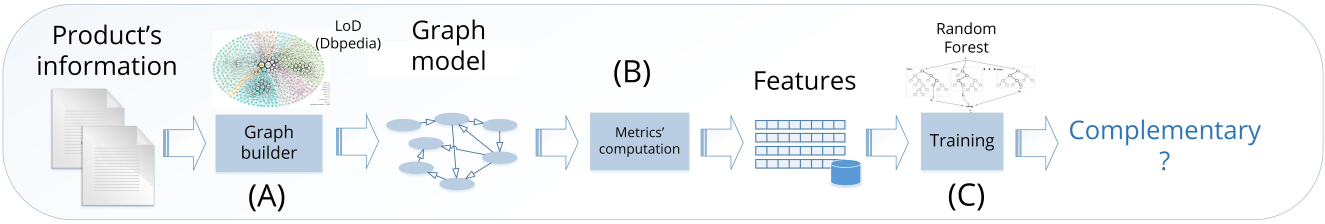

Figure 2: Workflow of *CompLoD*

Title: *Logitech M510 Wireless Mouse, Large Mouse, Computer Wireless Mouse.*
Category: *Electronics; Computers & Accessories; Computer Accessories & Peripherals; Keyboards, Mice & Accessories; Mice*
Description:
Contoured shape with soft rubber grips for all-day comfort
** Back/forward buttons and side-to-side scrolling plus zoom let you do more, faster Requires Logitech SetPoint software*
** 2-year battery life eliminates the need to replace batteries. Battery life may vary based on computing conditions*
** Comes with a tiny Logitech Unifying receiver that stays in your computer - plug it in, forget it*
** 3-year limited hardware warranty*
** Compatibility: Works with - Windows XP, Windows Vista, Windows 7, Mac OS X 10.4 or later, USB Port*
** Mouse Dimensions (height x width x depth): 2.56 in (65 mm) x 4.72 in (120 mm) x 1.61 in (41 mm)*

Example 1: Title, categories and description of a product (mouse)

As previously mentioned, attributes are extracted from the description of a product. Moreover, a category is a structured piece of information organized as a taxonomy. The categories go from generic (*Electronics*) to more specific (*Mice*). However, *CompLoD* doesn't make use of the taxonomy and sees all categories as independent.

Together all of this information is then processed in a pipeline-like fashion. First the product's meta-data is transformed into a structured graph in which product attributes and categories are matched against DBpedia resources (A). The resulting knowledge graph is a subgraph of DBpedia, extended with the products represented as synthetic nodes. Being aware that not all attributes have the same role in characterizing a product and that not all the interconnections are able to provide the same quality of information, we decided to attach scores to the nodes and paths in the graph using consolidated metrics (B). The results are used in the next stage to build the feature set for pairs of products. Finally, we train a classification model (C) and when this is fitted, the framework is able to answer the question whether one product is complementary to another one. More details about the individual tasks are provided in the following sections.

3.1 Building a knowledge graph

Our framework requires a knowledge graph as the starting point and we opted for DBpedia. In addition to the deterministic rules which are used to build the graph, DBpedia also has regularities with predictive power [20]. To make use of this, we need to integrate the products' meta-data into it by extracting structured information from the text. We therefore use *AlchemyAPI*, a NER tool. This identifies those tokens which correspond to published LoD IRIs. Whereas the tool supports the mapping to resources from several LoD datasets[4], we only store the information relative to DBpedia and leave the possibility of exploiting multiple datasets to future work. For instance, from the previous text some of the identified resources are[5]:

Categories
Electronics: *dbpedia:Electronics*
Computer & Accessories: *dbpedia:Computer, ...*
Attributes
Logitech: *dbpedia:Logitech, yago:Logitech, ...*
Operating system: *dbpedia:Operating_system, ...*
Microsoft Windows: *dbpedia:Microsoft_Windows*
Mac OS X: *dbpedia:Mac_OS_X, ...*
Microsoft: *dbpedia:Microsoft, yago:Microsoft, ...*
IMac: *dbpedia:IMac, yago:IMac, ...*
Windows Vista: *dbpedia:Windows_Vista, ...*

Example 2: Resources identified from the meta-data of the product showed in example 1

It can be observed that some of the identified resources correspond to categories, such as *dbpedia:Electronics*, while others are attributes, such as *dbpedia:Logitech* (brand). To keep the approach fully automated, we treat categories as text as well. The quality of this task highly depends on the efficacy of the NER tool, each of which has its own limitations. While most of the important resources are identified, some of them, like *dbpedia:Computer_mouse* (category), which are important to characterize the product, are not necessarily recognized. For more details about the expected performance of these tools we refer the reader to [8].

Given the products and the identified resources, the RDF graph is built as follows:

1. First we build a basic RDF graph G_p. We create one resource r_p for each product and connect it to its attributes IRIs. Let A_{r_p} be the bag[6] of attributes identified in the input text of the product represented by

[4]Currently supported to the date of publication are DBpedia, Freebase, US Census, GeoNames, UMBEL, OpenCyc, YAGO, MusicBrainz, CIA Factbook and CrunchBase.
[5]We use prefixes to shorten the namespaces:
dbpedia: http://dbpedia.org/resource/
yago: http://yago-knowledge.org/resource/
[6]The tokens can appear multiple times in the text. We have

r_p. We create a triple (r_p, sp, r_a) linking the product r_p with each $r_a \in A_{r_p}$ using a synthetic predicate sp. If an attribute r_a appears more than once in A_{r_p}, we create as many triples as the number of times r_a appears in A_{r_p}, where each triple has a different synthetic predicate.

2. Let C_{r_p} be the set of categories to which the product belongs. We also create triples (r_p, sp, r_c) that link products with their categories.

3. We enrich G_p by extracting a subgraph from DBpedia. This subgraph contains all DBpedia resources identified by the NER tool and all the resources reachable within 2 hops from them. We ignore the direction of the edges to do so. The reason why we limit the length to 2 is to reduce the computational cost of computing metrics on top of the graph (more details in the next section). We merge G_p with the extracted subgraph to form the *Extended graph* EG_p. Note that attributes and categories in EG_p are interconnected.

EG_p is then the input of the next task, in which metrics are computed to measure the importance of an attribute or a category for a product as well as the importance of the semantic paths interlinking them.

3.2 Computation of metrics

In our knowledge graph every product is connected to its attributes and categories. However, those attributes might have a distinct relevance in describing a product or even in describing a category (seen as a set of products). Therefore, we require some metrics that reflect the following intuition:

- How well can an attribute characterize a product, e.g. is *equalizer* more representative than *rechargeable* to describe a *speaker*? ***Section 3.2.1***.
- How well can an attribute characterize a category, e.g. is *4K* more representative than *wireless* for the category *Television & Video*? ***Section 3.2.2***.
- How relevant is the information carried by a path connecting two resources? Let's assume, for instance, that two product resources, a *television (tv)* and a *speaker (s)*, are connected through several paths of different lengths in EG_p.
  Is the path $tv \xrightarrow{\text{connectivity}} Bluetooth \xleftarrow{\text{connectivity}} s$ that goes through *Bluetooth* more important than another one that goes through *Stereo*? Which of these paths is the most informative? Does the length also play a role? Suppose that the connection is now
  $tv \xrightarrow{\text{connectivity}} Bluetooth \xrightarrow{\text{type}} Wireless \xleftarrow{\text{connectivity}} s$. Does the informativeness decay with the length of the path? ***Section 3.2.3***.

In order to provide answers to those questions and fulfill the requirements we employ *Term frequency-inverse document frequency* (TF-IDF)[19], a weighting scheme widely used in Information Retrieval and text mining to determine how important a term is to a document within a corpus. The design of such metrics is achieved through different instantiations of TF-IDF, i.e. we map different components of the RDF graph to terms and documents. This requires the consideration of a product r_p as a bag of attributes $\{r_{a_1}, r_{a_2}, ..., r_{a_k}\}$. A category r_c can be also thought of as a bag of attributes, namely those used to describe products of that category. We will explain these metrics in more detail in the following sections.

3.2.1 (Product) Attribute Frequency - Inverse Product Frequency (PAF-IPF)

The representation of a product as a bag of attributes allows us to define a weighting scheme that reflects how well an attribute r_a is able to represent or describe a product r_p. We instantiate TF-IDF using a product as a document and its attributes as the terms within the document.

$$PAF\text{-}IPF_{(r_a,r_p)} = PAF_{(r_a,r_p)} \times IPF_{(r_a)}$$
$$IPF_{(r_a)} = \log \frac{|P|}{PF_{(r_a)}}$$

Intuitively, $PAF_{(r_a,r_p)}$ counts the number of times a product r_p is described by an attribute r_a, i.e. it counts the number of triples that link the product to an attribute. In a similar way $PF_{(r_a)}$ counts the number of products that have r_a as an attribute. $|P|$ is the cardinality of the set P, i.e. the number of products available in G. The goal of $IPF_{(r_a)}$ is to reduce the score by a factor which is proportional to the number of times the attribute is used to describe products. If an attribute is common to all products then it cannot strongly characterize the product.

3.2.2 (Category) Attribute Frequency - Inverse Category Frequency (CAF-ICF)

In the same way we define the following:

$$CAF\text{-}ICF_{(r_a,r_c)} = CAF_{(r_a,r_c)} \times ICF_{(r_a)}$$
$$ICF_{(r_a)} = \log \frac{|C|}{CF_{(r_a)}}$$

which reflects to which extent an attribute characterizes a category. $CAF_{(r_a,r_c)}$ counts the number of times that an attribute resource r_a is used to describe a product r_p that belongs to a category r_c. $CF_{(r_a)}$ counts the number of categories in which the attribute r_a is used to describe at least one product which belongs to that category. $|C|$ is the cardinality of the set C. The role of $ICF_{(r_a)}$ is the same as that of $IPF_{(r_a)}$.

3.2.3 Path Informativeness

In addition to the two metrics introduced above, we require another one which reflects the degree of informativeness a path carries with it. We use the definition of path informativeness presented in [26], which we tailored for our needs. This concept is based on a similar instantiation of TF-IDF. RDF resources r are used as documents. However, two kinds of documents are considered: those in which r appears as a subject of a triple and those in which it has the role of an object. This allows us to define the following metric:
Predicate frequency - Inverse Triple Frequency PF-ITF:

$$I\text{-}PF\text{-}ITF_{(p,r)} = I\text{-}PF_{(p,r)} \times ITF_{(p)}$$
$$O\text{-}PF\text{-}ITF_{(p,r)} = O\text{-}PF_{(p,r)} \times ITF_{(p)}$$
$$ITF_{(p)} = \log \frac{|T|}{|T(p)|}$$

to preserve the information of how often the attribute occurs in order to compute metrics in the next stage.

I-PF-ITF$_{(p,r)}$ is the *Input PF-ITF* of a term p within the document resource r and *O-PF-ITF*$_{(p,r)}$ is the *Output PF-ITF* of a term p within the document resource r. *I-PF*$_{(p,r)}$ counts the number of resources that can be mapped to the variable $?x$ of the triple pattern $(?x, p, r)$. This is also the case for $O\text{-}PF(p, r)$ in which the possible mappings for $?y$ in $(r, p, ?y)$ are counted. $|T|$ is the overall number of triples whereas $|T(p)|$ is the number of triples in which p appears. The role of $ITF_{(p)}$ is the same as for the previous instantiations.

This metric allows us to define the informativeness metric. For a path $\rho(r_i, r_j)_{(p)}$ of length 1 the informativeness $\mathcal{I}$ is defined as:

$$\mathcal{I}_{\rho(r_i,r_j)_{(p)}} = \frac{\textit{O-PF-ITF}_{(p,r_i)} + \textit{I-PF-ITF}_{(p,r_j)}}{2}$$

For paths of length t the informativeness is defined as the sum of the informativeness of its components divided by the length of the path.

$$\rho(r_i, r_j)_{(p1,\ldots,p_{t-1},p_t)}$$
$$\mathcal{I}_{\rho(r_i,r_j)_{(p1,\ldots,p_{t-1},p_t)}} = (\mathcal{I}_{\rho(r_i,r_k)_{(p1)}} \ + \ ... \ + \ \mathcal{I}_{\rho(r_q,r_s)_{(p_{t-1})}} \ + \ \mathcal{I}_{\rho(r_s,r_j)_{(p_t)}})/t$$

Given the previous definitions, it is possible to define $\mathcal{I}_{max}$, i.e. the maximum informativeness carried by a path connecting two resources r_i and r_j.

$$\mathcal{I}_{max}(r_i, r_j) = max\{\mathcal{I}(r_i, r_j)_{\rho_1}, ..., \mathcal{I}(r_i, r_j)_{\rho_k}\},$$

where ρ_i is an arbitrary path. Each pair of paths ρ_i and ρ_j is different, i.e. there exists at least one predicate in both paths which differs. Similarly, it is possible to define $\mathcal{I}_{avg}$, the average informativeness of all paths between r_i and r_j.

$$\mathcal{I}_{avg}(r_i, r_j) = (\mathcal{I}(r_i, r_j)_{\rho_1} + ... + \mathcal{I}(r_i, r_j)_{\rho_k})/k$$

To summarize. The metrics allow us to determine the importance of attributes in describing a product or a category, or to measure the amount of informativeness carried by the semantic paths connecting two resources. These are designed considering different granularities for the definition of *document*. In the case of PAF-IPF a document is a product and the terms are the attributes that describe the product. In the second case, CAF-ICF, a document is a category and the terms are the attributes used to describe products of that category. In the case of informativeness documents are subject or object resources whereas predicates are the terms. *CompLoD* computes the relevance metrics in EG_p by iterating over the set of identified attributes and then it proceeds by searching paths linking between them.
All these metrics have been validated by other works and have been used to solve a variety of problems. However, when one focuses again on the problem illustrated in figure 1, i.e. the problem of designing a feature set that applies to pairs of products, one can notice that these metrics cannot be used directly as features. For instance, PAF-IPF and CAF-ICF are metrics which apply to a single product. While nothing prevents one from using them as features for pairs, it is not possible to have each possible attribute as a feature. This would lead to an extremely high-dimensional feature vector. And whereas the path informativeness between two products' attributes could be thought of as a feature for pairs of products, one cannot have each possible path as a feature either, for the same reason explained above (high-dimensionality).

We propose therefore a strategy to combine these metrics, by following a very natural intuition which turned out to produce useful results. This allows one to design a low-dimensional feature vector which can be used to model pairs of products.

3.3 Features engineering

To the best of our knowledge there is no work which addresses the issue of which features for pairs of products (r_{p_i}, r_{p_j}) can be used to predict the *complementary* relationship. We aim at filling this gap. Our rationale is based on the observation that for products belonging to different domains there might exist correspondences and dependencies between their attributes. This has been validated in marketing, behavioral, and data mining research studies [4]. Our hypothesis is that (1) the correspondences and dependencies tell us also something about the *complementary* relationship, and (2) this correspondence can be measured by the informativeness carried in the interconnection between attributes, weighted by those attributes' relative importance. We therefore explain in this section how to leverage the interconnections of the knowledge graph and to combine the metric scores in a meaningful way.

Let $A_{r_{p_i}}$ and $A_{r_{p_j}}$ be the bag of attributes of products r_{p_i} and r_{p_j}. To simplify the explanation of the formulas to calculate the features we will from now on assume that r_{a_i} and r_{a_j} are attributes of r_{p_i} and r_{p_j}, respectively. Moreover, we also assume that r_{c_i} and r_{c_j} are the most specific categories of r_{p_i} and r_{p_j}. For the sake of readability we introduce two weighting functions ω_P and ω_C shown in figure 3, which use PAF-IPF and CAF-ICF, respectively. In this way, we can weigh a path between two attributes of r_{a_i} and r_{a_j} according to their characterization relevance. Next, we will present the design choices we made.

Maximal Informativeness over the most relevant product attributes (*MIMPA*): this feature combines PAF-IPF and the concept of informativeness as follows. Given a product r_{p_i}, we first search for the attribute r_{a_i} that has the maximum PAF-IPF$_{(r_{a_i},r_{p_i})}$ among other attributes. We do the same for r_{p_j}. Let $r_{a_i}^{M_p}$ and $r_{a_j}^{M_p}$ be these attributes for r_{p_i} and r_{p_j}, respectively. These are the most relevant product attributes. The feature is then designed as follows:

$$MIMPA_{(r_{p_i},r_{p_j})} = \mathcal{I}_{max}\left(r_{a_i}^{M_p}, r_{a_j}^{M_p}\right) \cdot \omega_P(r_{a_i}^{M_p}, r_{a_j}^{M_p})$$

Recall that when computing $\mathcal{I}_{max}(r_{a_i}^{M_p}, r_{a_j}^{M_p})$ all possible paths which connect $r_{a_i}^{M_p}$ to $r_{a_j}^{M_p}$ are considered and from these, it returns the max. informativeness value.

Maximal Informativeness over the most relevant category attributes (*MIMCA*): we focus again on the attribute bags of both products. However, rather than searching for those which characterize the products the best, we take a look to the most specific category of each product and leverage CAF-ICF to find the attribute which characterizes this category the best. Let $r_{a_i}^{M_c}$ and $r_{a_j}^{M_c}$ be these attributes for r_{p_i} and r_{p_j}, respectively. Then:

$$MIMCA(r_{p_i}, r_{p_j}) = \mathcal{I}_{max}\left(r_{a_i}^{M_c}, r_{a_j}^{M_c}\right) \cdot \omega_C(r_{a_i}^{M_c}, r_{a_j}^{M_c})$$

$$\omega_P(r_{a_i}, r_{a_j}) = \sqrt{\frac{\left(\text{PAF-IPF}_{(r_{a_i}, r_{p_i})}\right)^2 + \left(\text{PAF-IPF}_{(r_{a_j}, r_{p_j})}\right)^2}{2}}, \qquad \omega_C(r_{a_i}, r_{a_j}) = \sqrt{\frac{\left(\text{CAF-ICF}_{(r_{a_i}, r_{c_i})}\right)^2 + \left(\text{CAF-ICF}_{(r_{a_j}, r_{c_j})}\right)^2}{2}}$$

Figure 3: Weighting functions ω_P and ω_C

where r_{c_i} and r_{c_j} are the categories of products r_{p_i} and r_{p_j}, respectively.

Average Informativeness over the most relevant product attributes (*AIMPA*): this feature works in a similar way as MIMPA. When the most relevant attributes of the products, $r_{a_i}^{M_p}$ and $r_{a_j}^{M_p}$, are identified, all paths connecting these resources are considered. Therefore, the average informativeness $\mathcal{I}_{avg}$ is considered and not only the score of the most informative one.

$$AIMPA(r_{p_i}, r_{p_j}) = \mathcal{I}_{avg}\left(r_{a_i}^{M_p}, r_{a_j}^{M_p}\right) \cdot \omega_P(r_{a_i}^{M_p}, r_{a_j}^{M_p})$$

Average Informativeness over the most relevant category attributes (*AIMCA*): in the same way as MIMCA, we consider here the attributes which characterize the category the most, but we use $\mathcal{I}_{avg}$ instead of $\mathcal{I}_{max}$, i.e. we consider the average informativeness of all paths connecting them.

$$AIMCA(r_{p_i}, r_{p_j}) = \mathcal{I}_{avg}\left(r_{a_i}^{M_c}, r_{a_j}^{M_c}\right) \cdot \omega_C(r_{a_i}^{M_c}, r_{a_j}^{M_c})$$

Average of maximal informativeness over all attributes (*AMIO*): rather than considering the most relevant attribute for each product, we focus here on the informativeness flowing from one product (from all of its attributes) to the other one. To do so we consider all pairs of attributes r_{a_i} and r_{a_j}, which belong to r_{p_i} and r_{p_j}, respectively, and compute the maximum informativeness carried by the links of each product pair. As for previous features we also combine the relevance of individual attributes with the informativeness of the link.
Let $P_{(r_{p_i}, r_{p_j})} = \{(r_{a_{i1}}, r_{a_{j1}}), (r_{a_{i1}}, r_{a_{j2}}), ..., (r_{a_{ik}}, r_{a_{jl}})\}$ be a set of attributes tuples, where the first element is an attribute of r_{p_i} and the second of r_{p_j}. Then:

$$AMIO_P(r_{p_i}, r_{p_j}) = \frac{1}{\mid P_{(r_{p_i}, r_{p_j})} \mid} \cdot \left(\sum_{(r_{a_i}, r_{a_j}) \in P_{(r_{p_i}, r_{p_j})}} \mathcal{I}_{max}(r_{a_i}, r_{a_j}) \cdot \omega_P(r_{a_i}, r_{a_j}) \right)$$

$$AMIO_C(r_{p_i}, r_{p_j}) = \frac{1}{\mid P_{(r_{p_i}, r_{p_j})} \mid} \cdot \left(\sum_{(r_{a_i}, r_{a_j}) \in P_{(r_{p_i}, r_{p_j})}} \mathcal{I}_{max}(r_{a_i}, r_{a_j}) \cdot \omega_C(r_{a_i}, r_{a_j} \right)$$

$$AMIO(r_{p_i}, r_{p_j}) = \frac{AMIO_P(r_{p_i}, r_{p_j}) + AMIO_C(r_{p_i}, r_{p_j})}{2}$$

AMIO combines both, the characterization power of the attributes in the role of product attributes and category attributes.

Average of average informativeness over all attributes (*AAIO*): this feature is computed in the same way as AMIO, except that $\mathcal{I}_{avg}$ is used instead of $\mathcal{I}_{max}$. In this way we consider the complete flow of informativeness between each pair of attributes.

Maximal relevance of common product attributes (*MRCPA*): when considering only the common attributes of both products $A_{r_{p_i}} \bigcap A_{r_{p_j}}$, we compute the PAF-IPF of the elements of the intersection and return the maximum value.

$$MRCPA(r_{p_i}, r_{p_j}) = max\ \{\ \omega_P(r_{a_i}, r_{a_j}) : r_{a_i} = r_{a_j} \in (A_{r_{p_i}} \cap A_{r_{p_j}})\}$$

Maximal relevance of common category attributes (*MRCCA*): this is the symmetrical counterpart of MRCPA in which one considers the contribution of common attributes in characterizing the product categories.

$$MRCCA(r_{p_i}, r_{p_j}) = max\ \{\ \omega_C(r_{a_i}, r_{a_j}) : r_{a_i} = r_{a_j} \in (A_{r_{p_i}} \cap A_{r_{p_j}})\}$$

Main product relevance (*MPR*): this metric measures the relevance of the main product (first element of the pair at the input sample) on the basis of its attributes. Let $|(r_a, ?p, ?o)|$ be the number of triples having r_a as subject and $|(?s, ?p, r_a)|$ be the number of triples having r_a as object. Then:

$$MPR(r_{p_i}) = \sum_{r_a \in A_{r_{p_i}}} (|(r_a, ?p, ?o)| + |(?s, ?p, r_a)|)$$

Related product relevance (*RPR*): this metric is the same as the previous one, but applied to the related product (second element of the pair at the input sample).
Interconnection between products (*IP*): this feature aims at reflecting the degree of connectivity between the attributes of two products. Let φ be the number of attributes pairs $(r_{a_i}, r_{a_j}) \in P_{(r_{p_i}, r_{p_j})}$ in which there exist at least one path connecting them. Then:

$$IP(r_{p_i}, r_{p_j}) = \frac{\varphi}{|P_{(r_{p_i}, r_{p_j})}|}$$

Product similarity (*PS*): to measure the degree of similarity, we use the Jaccard similarity to the set of attributes of both products:

$$PS(r_{p_i}, r_{p_j}) = \frac{|A_{r_{p_i}} \bigcap A_{r_{p_j}}|}{|A_{r_{p_i}} \bigcup A_{r_{p_j}}|}$$

Other metrics: We designed and tried several other features whose contribution was small or minimal.

In this section we introduced the features which played an important role in predicting whether one product is complementary to another one. All the 12 features are the result

of a feature selection task performed to boost the accuracy prediction of the classifiers. These are shown on top of the next page together with their relative importance[7].

3.4 Classification model

With enough input samples and a well-defined feature set the last missing step consists of fitting a classifier. A wide range of classification techniques exist for supervised learning. We compared the performance of different kinds of classifiers, ensemble and non, and random forest was the fittest model. Moreover, we validated our model and carried out hyper-parameter optimization. More details can be found in the next section.

4. EXPERIMENTS

We conducted experiments to assess the performance of *CompLoD*. Some of these experiments helped us to choose the most suitable classifier and to optimize it. But the central question, whether the designed feature set was really significant, could be also answered. Each of the single tasks of *CompLoD* were run on a single machine with a processor Intel(R) Xeon(R) CPU E3-1231@3.80GHz with 4 cores and 32 GB of RAM.

Dataset. We used a dataset from Amazon.com[8]. This contains products' meta-data for each category. Our experiments focus on the category *Electronics*, which includes a wide range of subcategories, such as *Internal Hard Drives*, *eBook Readers* or *Camera Batteries*, etc. The overall number of subcategories considered is 817 whereas the number of products is ca. 0.5 million. In addition to the meta-data for each product the dataset also contains four lists of related products: *also bought*, *also viewed*, *bought together*, *buy after viewing*. As in [14] we assume that *also bought* and *bought together* products are complementary. Products from the remaining lists are substitutable products[9]. We use them as the negative class (non-complementary). We manually verified that this assumption is not always correct: for some products some related products appeared in both groups. Assuming that a product cannot be both, substitutable and complementary to another one, we removed those ambiguous related products. As a result of this process, 11% of complementary products and 8% of non-complementary products were removed from the dataset.

Building the graph. One of the initial stages of the framework consists of identifying from the text those tokens which correspond to resources published in the LoD cloud. Approx. 3500 resources were recognized by the NER tool, which is a relatively low number. However, for 96% of the products at least one resource could be identified. The unmatched 4% was removed from the dataset. By querying DBpedia's SPARQL remote endpoint we extract the triples in which those resources appear as subject or object of a triple. We repeat this process iteratively until we obtain all resources reachable within 2 hops from the identified resources. The resulting graph has approx. 1.4 million resources.

[7]The importance is computed as the normalized total reduction of the criterion represented by that feature (Gini importance of random forests).

[8]The dataset is part of the project "Stanford Network Analysis Project (SNAP)" and made available for research.

[9]We are not addressing in this work the task of predicting the substitutable relationship.

Computing metrics. Once the knowledge graph is built we compute the metrics explained in section 3.2. The PAF-ICF and CAF-ICF relevance scores and the informativeness of the paths are efficiently stored in the system using inverted indexes for easy retrieval.

Classification. With the metrics computed, we compute the features as explained in section 3.3 for those pairs of products which appear in the ground truth. We store the feature vectors and attach the class label to it, *complementary* and *non-complementary*. The overall number of input samples is ca. 3.7 million, of which 1.4 million represents complementary pairs and 2.3 million pairs the non-complementary counterpart. These input samples were then split into training and test sets. We kept the distribution at 50% of positive and 50% negative samples. We started with a relatively small number of samples for the training set (10%) and we gradually increased it until we reached 90%. In this way we were able to analyze the learning rate of the classifiers. We tried the performance of several of them, such as *Decision Trees, Random Forest, AdaBoost, Naive Bayes (Gaussian), Linear Discriminant Analysis* and *Quadratic Discriminant Analysis*. We used the following rates computed from the confusion matrix (*TP*=true positive, *FP*=false positive, *TN*=true negative, *FN*=false negative):

$$Precision(+) = \frac{TP}{TP+FP}; \quad Precision(-) = \frac{TN}{TN+FN}$$

$$Recall(+) = \frac{TP}{TP+FN}; \quad Recall(-) = \frac{TN}{TN+FP}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

$$F1\text{-}Measure(+) = 2 \cdot \frac{Precision(+) \cdot Recall(+)}{Precision(+) \cdot Recall(+)}$$

The label (+) in the metric focuses on the quality attribute for predicting the positive class whereas (-) focuses on the negative one. Accuracy is then defined as the mean of the positive and negative precision score. The performance of the different classifiers was similar. To avoid overfitting and make sure that the model generalizes to an independent dataset, we validate our model with K-Fold and Monte Carlo cross-validation (CV). While k-Fold CV makes sure that each of the folds is used at least once as a test set, Monte Carlo CV randomly shuffles the whole dataset, thus generating independent partitions for each run. The final results are shown in Figure 4(A), which illustrates that the Random Forest classifier improves its accuracy with more input samples from which it can learn, being able to obtain up to ca. 78.5% accuracy. In the same figure (B) shows that the recall for the negative class overtakes that of the positive class. It is important to mention that a TN refers to a non-complementary ground truth pair which is predicted to be non-complementary. As the very last task we performed hyper-parameter optimization by exploiting two techniques, namely Random Search (run with 30 iterations) and Grid Search (run with the top-3 parameter set).

Results and reflections. One of the unique advantages of our framework is that new products have a chance to be classified as complementary to other products **even in the absence of previous purchase history of the product**. The results are certainly not as good as for the case for which transaction data or post-transaction feedback is

MIMPA	MIMCA	AIMPA	AIMCA	AMIO	AAIO	MRCPA	MRCCA	MPR	RPR	IP	PS	class
6.87	10.32	8.27	11.48	5.73	9.01	6.89	10.25	9.75	9.17	5.04	7.17	1 or 0

Feature set for pairs of product resources (r_{p_i}, r_{p_j}) and their feature importance.

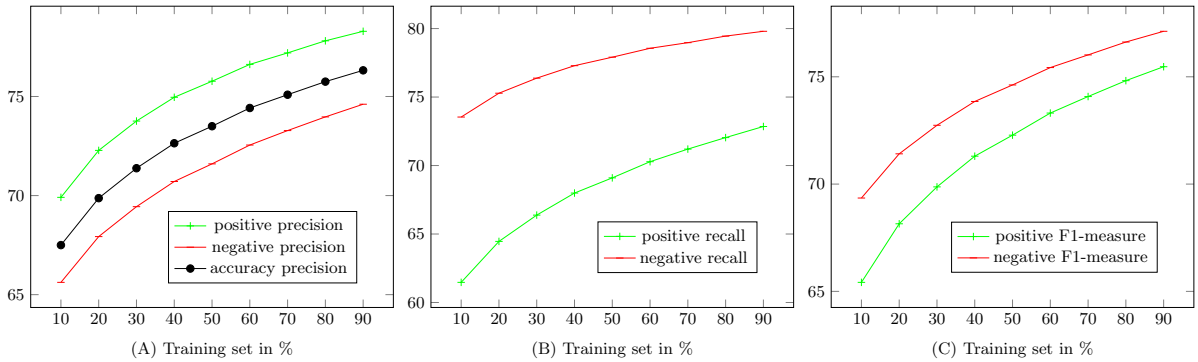

Figure 4: Performance results of *CompLoD* in terms of (A) accuracy, (B) recall, and (C) F1-measure

available, but this work should open a new door for further contributions. For instance, the work of McAuley et al. in [14], which uses post-transaction feedback to learn a model, obtained an accuracy of 88% on the same dataset and category using reviews and additional meta-data which we didn't consider in our work, e.g. prices[10]. The performance of *CompLoD* is then surprisingly good at taking into account that only minimal information about the product is used, therefore making it well-suited to deal with the cold-start problem.

CompLoD also enables a different perspective. Whenever the relationship between two products is identified as *complementary* and an explanation is required, it is possible to go back to the graph components and the values of the computed graph metrics from the values of the single features and to provide the user with a reasonable explanation. For instance, one could use the most relevant attributes or category attributes of both products, e.g. those that maximize PAF-IPF or CAF-ICF, respectively, and the path connecting those attributes which maximizes the informativeness. Figure 5 illustrates this. All these components are a valuable source of explanation. However, assessing which of these graph components are the most useful would require an online study with real users.

Computational cost and scalability. PAF-IPF and CAF-ICF have a cost of $\mathcal{O}(|P|+|A|)$ and $\mathcal{O}(|C|+|A|)$, respectively. The cost of path informativeness is higher: $\mathcal{O}(|I|+|Pred|)$, where *Pred* is the set of all vocabulary predicates. While it is obvious that extracting information from a large graph and computing metrics over it is certainly expensive, most of the tasks done in the single stages of the framework can be pre-computed off-line. For instance, one could try to keep as many attributes in the graph as possible or even try to anticipate attributes of soon-to-be-released products. The metrics can be then computed and stored at once. In this way when a new product arrives in the system, all of its attributes will already be present in the graph, together with the informativeness scores of the semantic paths connecting them to other attributes.

[10]The results are not comparable only in terms of the learning source but also on the fact that they only considered in their experiments products with 20 reviews or more, which means that only 20% of the pairs of products contained in the original ground truth remain after filtering those products out.

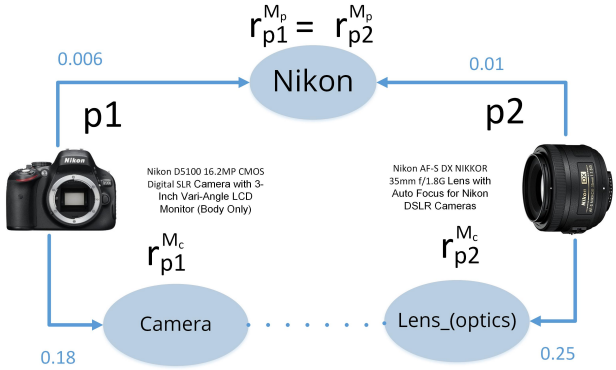

Figure 5: Two products and their most relevant attributes

5. RELATED WORK

Approaches to find complementary products have a wide range of applications, from product placement to products bundling to website structure optimization, etc. In particular, on-line retailers make use of complementary products to provide recommendations [30, 35], which help the users to discover related products while improving their volume of sales. Most of these techniques typically consist of mining transactional purchase data, event logs, or users' implicit feedback. However, for new products for which such information is not available or scarce, most of these approaches are not applicable or fail altogether.

As in [14], we aim at *learning* the semantics of this relationship, thus making it possible to disregard transactions. Therefore, we build a knowledge graph in which products,

attributes, and categories are represented as nodes interconnected by semantic paths. Knowledge acquisition is in general a very costly task. Therefore, we extract this information from DBpedia, whose usage as a knowledge base has been validated in other fields, such as that of Cross-domain Recommender Systems [6, 9, 23, 29].

Being able to learn from the semantics encoded in the graph and the patterns therein has some commonalities with *Link Discovery* or *Link prediction*. These approaches see a knowledge graph as a statistical model. The existence of triples is represented in these models as binary random variables which are correlated with each other. Different assumptions for the correlation lead to different models, e.g. based on latent features [21], graph features or those in which variables have local interactions [20]. Our work is closer to observable graph feature models, in the sense that some of the problems addressed there, e.g. designing a feature set from a graph to fit a learning model, are similar. For instance, [1] focuses on predicting future friendship links in *Social Networks* using topological features. Aiming at the same goal, [5] presents and uses a combination of a large number of features. These techniques are not necessarily limited to Social Networks, but they are used in fields like Biomedicine [10] or Genomics [34].

However, there is a central difference between these approaches and ours. First, we deal here with a knowledge graph in which the structured information is integrated into a complex ontology. This means two nodes can be interconnected in several ways which are not necessarily known in advance. This differs from the kinds of graphs typically treated in the link prediction problem, e.g. social, concept, or interaction networks, whose edges intend rather to represent a small number of relationships and are therefore less flexible in the representation. Secondly, many of the features proposed are based on local similarity indices, such as Common Neighbors, Adamic-Adar Index, Preferential Attachment Index, Jaccard Index, etc. and might therefore be too localized to capture the patterns which determine the complementary relationship.

A related problem can also be found in the field of online search behavior [16]. In this area some efforts have been done to predict whether two queries posed by a user within a single or multiple sessions aim at performing the same task. To do so, some approaches try to model hierarchies of tasks and subtasks. For example, booking a hotel and registering for a conference venue are subtasks of planning a conference attendance. These methods are typically based on classification, as in our case, or clustering [11, 12, 17, 33]. Since a purchase of one product and a complementary one can be thought of as the problem of matching tasks, we consider this line of research related work.

6. CONCLUSION AND FUTURE WORK

We presented *CompLoD*, a framework that leverages different models and techniques to identify complementary products while only using a product's available meta-data, such as titles, descriptions and categories. By extracting structured information from the text, we manage to build a knowledge graph where products' attributes are interconnected by semantic paths. Being able to identify complementary products from this graph rather than using transactions, as most of the current techniques do, solves the cold-start problem for new items. Moreover, the fact that this graph is based on DBpedia demonstrates the usefulness and value of Semantic Web standards. The prediction task in the last stage is implemented with a Random Forest Classifier which predicts, for a pair of products, whether one product is complementary to the other. However, such a model requires a feature set which represents properties of pairs of products. To the best of our knowledge, this is the first time where the problem of extracting this information from a semantic graph to predict the complementary relationship is addressed. Our experiments show that the classifier is able to learn the relationship from our designed feature set, thus validating it.

Although the predictive power of our system is surprisingly good, given that this is achieved only by using products' meta-data, it is important to mention that the quality of *CompLoD* depends on the quality of the models and tools used in the single stages. This dependency consequently leaves room for improvement. Therefore, we would like to try different NER tools and LoD datasets and investigate the qualitative impact. We could also exploit a more domain-specific LoD dataset. For this reason, the framework is extensible, i.e. all components can be extended or replaced.

In the future we would like to conduct further experiments considering other categories, to strengthen the significance of our approach. Predicting whether a product is complementary to another one might still produce too many recommendation candidates for a single user. Therefore, we would like to extend our model to be able to further refine the list of complementary products taking into account the user's taste. For this task, a ranking strategy might be more suitable than a classification one. We would also like to include features that reflect the hierarchical structure of categories and in general to continue searching for new features that improve the accuracy of the system.

7. REFERENCES

[1] C. Ahmed, A. ElKorany, and R. Bahgat. A supervised learning approach to link prediction in twitter. *Social Network Analysis and Mining*, 6(1):24, 2016.

[2] A. Aribarg and N. Z. Foutz. Category-based screening in choice of complementary products. *Journal of Marketing Research*, 46(4):518–530, 2009.

[3] A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub, and I. Netanely. Movie recommender system for profit maximization. In *7th ACM Conf. on Recommender Systems, RecSys '13, Hong Kong, China*, pages 121–128, 2013.

[4] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. Cross-domain recommender systems. In *Recommender Systems Handbook*, pages 919–959. 2015.

[5] W. Cukierski, B. Hamner, and B. Yang. Graph-based features for supervised link prediction. In *Neural Networks (IJCNN), Int. Joint Conf. on*, pages 1237–1244, July 2011.

[6] I. Fernández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci. A generic semantic-based framework for cross-domain recommendation. In *Proc. of the 2Nd Int. Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '11, pages 25–32, 2011.

[7] F. Gedikli and D. Jannach. Neighborhood-restricted mining and weighted application of association rules

for recommenders. In *Proc. of the 11th Int. Conf. on Web Information Systems Engineering*, WISE'10, pages 157–165, 2010.

[8] T. Heuss, B. Humm, C. Henninger, and T. Rippl. A comparison of ner tools w.r.t. a domain-specific vocabulary. In *Proc. of the 10th Int. Conf. on Semantic Systems*, SEM '14, pages 100–107, 2014.

[9] M. Kaminskas, I. Fernández-Tobías, I. Cantador, and F. Ricci. *Ontology-Based Identification of Music for Places*, pages 436–447. 2013.

[10] J. Katukuri, Y. Xiey, V. V. Raghavan, and A. Gupta. Supervised link discovery on large-scale biomedical concept networks. In *Proc. of the 2011 IEEE Int. Conf. on Bioinformatics and Biomedicine*, BIBM '11, pages 562–568, 2011.

[11] L. Li, H. Deng, A. Dong, Y. Chang, and H. Zha. Identifying and labeling search tasks via query-based hawkes processes. In *Proc. of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD '14, pages 731–740, 2014.

[12] Z. Liao, Y. Song, Y. Huang, L. w. He, and Q. He. Task trail: An effective segmentation of user search behavior. *IEEE Transactions on Knowledge and Data Engineering*, 26(12):3090–3102, Dec 2014.

[13] F. Manola, E. Miller, and B. McBride. RDF Primer. W3C Recom., 2004.

[14] J. J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *Proc. of the 21th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 2015*, pages 785–794, 2015.

[15] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *Proc. of the 38th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, Santiago, Chile, 2015*, pages 43–52, 2015.

[16] R. Mehrotra, P. Bhattacharya, and E. Yilmaz. Uncovering task based behavioral heterogeneities in online search behavior. In *Proc. of the 39th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, SIGIR '16, pages 1049–1052, 2016.

[17] R. Mehrotra and E. Yilmaz. Towards hierarchies of search tasks & subtasks. In *Proc. of the 24th Int. Conf. on World Wide Web*, WWW '15 Companion, pages 73–74, 2015.

[18] R. Meymandpour and J. G. Davis. Recommendations using linked data. In *Proc. of the 5th Ph.D. Workshop on Information and Knowledge Management, PIKM 2012, Maui, HI, USA, 2012*, pages 75–82, 2012.

[19] I. C. Mogotsi. Christopher d. manning, prabhakar raghavan, and hinrich schütze: Introduction to information retrieval - cambridge university press, cambridge, england, 2008, 482 pp, ISBN: 978-0-521-86571-5. *Inf. Retr.*, 13(2):192–195, 2010.

[20] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. of the IEEE*, 104(1):11–33, Jan 2016.

[21] M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing yago: Scalable machine learning for linked data. In *Proc. of the 21st Int. Conf. on World Wide Web*, WWW '12, pages 271–280, 2012.

[22] D. Nikovski and V. Kulev. Induction of compact decision trees for personalized recommendation. In *Proc. of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 575–581, 2006.

[23] V. C. Ostuni, T. Di Noia, E. Di Sciascio, S. Oramas, and X. Serra. A semantic hybrid approach for sound recommendation. In *Proc. of the 24th Int. Conf. on World Wide Web*, WWW '15 Companion, pages 85–86, 2015.

[24] B. Pathak, R. Garfinkel, R. D. Gopal, R. Venkatesan, and F. Yin. Empirical analysis of the impact of recommender systems on sales. *Journal of Management Information Systems*, 27(2):159–188, 2010.

[25] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of SPARQL. *ACM Trans. Database Syst.*, 34(3), 2009.

[26] G. Pirrò. Reword: Semantic relatedness in the web of data. In *Proc. of the 26th AAAI Conf. on Artificial Intelligence, 2012, Toronto, Ontario, Canada.*, 2012.

[27] T. Raeder and N. V. Chawla. Market basket analysis with networks. *Social Network Analysis and Mining*, 1(2):97–113, 2011.

[28] F. Ricci, L. Rokach, and B. Shapira, editors. *Recommender Systems Handbook*. 2015.

[29] P. Ristoski, M. Schuhmacher, and H. Paulheim. Using graph metrics for linked open data enabled recommender systems. In *E-Commerce and Web Technologies - 16th Int. Conf. on Electronic Commerce and Web Technologies, EC-Web 2015, Valencia, Spain, 2015, Revised Selected Papers*, pages 30–41, 2015.

[30] A. Sharma, J. M. Hofman, and D. J. Watts. Estimating the causal impact of recommendation systems from observational data. In *Proc. of the 16th ACM Conf. on Economics and Computation*, EC '15, pages 453–470, 2015.

[31] G. Suchacka and G. Chodak. Using association rules to assess purchase probability in online stores. *Information Systems and e-Business Management*, pages 1–30, 2016.

[32] I. M. A. O. Swesi, A. A. Bakar, and A. S. A. Kadir. Mining positive and negative association rules from interesting frequent and infrequent itemsets. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th Int. Conf. on*, pages 650–655, May 2012.

[33] H. Wang, Y. Song, M.-W. Chang, X. He, R. W. White, and W. Chu. Learning to extract cross-session search tasks. In *Proc. of the 22Nd Int. Conf. on World Wide Web*, WWW '13, pages 1353–1364, 2013.

[34] Z. You, S. Zhang, and L. Li. Integration of genomic and proteomic data to predict synthetic genetic interactions using semi-supervised learning. In *Proc. of the 5th Int. Conf. on Emerging Intelligent Computing Technology and Applications*, ICIC '09, pages 635–644, 2009.

[35] J. Zheng, X. Wu, J. Niu, and A. Bolivar. Substitutes or complements: Another step forward in recommendations. In *Proc. of the 10th ACM Conf. on Electronic Commerce*, EC '09, pages 139–146, 2009.